"the packet"

The newsletter of V.A.D.C.G.

The Vancouver Amateur Digital Communications Group

Issue 6 December 1981

CONTENTS

Vancouver Amateur Digital Communications Group
818 Rondeau Street, Coquitlam, British Columbia, Canada V3J5Z3

```
        TITLE   'VADCG TNC - MODULE TIP-TTC (LAST REVISED 1500 31-AUG-81)'
; ***************************************************************************
; **    VADCG TERMINAL NODE COMMUNICATIONS PROGRAM  -  MODULE TIP-TTC      **
; **          BY DOUG LOCKHART, VE7APU                    MAY, 1980        **
; ***************************************************************************
; LAST CHANGED: JULY 13, 1981

; TERMINAL INTERFACE PROGRAM
; THIS PROGRAM IS WRITTEN TO RUN IN THE VADCG TERMINAL NODE CONTROLLER.  IT
; INTERFACES WITH A NODE COMMUNICATIONS PROGRAM RUNNING AT ADDRESS 0 IN
; MEMORY.  THIS VERSION IS WRITTEN TO USE THE 8250 PROGRAMMABLE UART
; TO COMMUNICATE WITH A COMPUTER.
; THE BASIC FEATURES OF THIS TIP ARE:
; NO ECHO OF DATA TO DIGITAL EQUIPMENT
; AUTOMATIC TRANSMISSION OF DATA IN BUFFER AFTER DELAY
; AUTOMATIC PACKET GENERATION IF DATA IS RECEIVED WITHOUT LF.
; CTS FLOW CONTROL FROM DIGITAL EQUIPMENT TO TIP
;
;       WHEN DSR IS HIGH, CONNECTIONS ARE ALLOWED. WHEN IT FALLS, A
;       DISCONNECT IS DONE. DTR IS SET HIGH IF CONNECTED, LOW IF NOT
;
;       TRANSMIT INTERRUPT CODE HAS BEEN RE-WRITTEN AND INCLUDED
;       IN THE DISPATCHER AT MAINLINE LEVEL TO ASSURE CORRECT
;       SYNCHRONIZING OF THE HOST. WHEN CONNECTED, EACH TIME THE
;       CTS LINE RISES, THE TIP MAY SEND 1 BYTE BUT MUST WAIT
;       TILL IT FALLS BEFORE RE-ARMING.
;
        MACLIB  LIB85   ; INCLUDE EXTRA 8085 INSTRUCTION SET

INCTB   MACRO   ?D
        IF      NOT NUL ?D
        MVI     A,?D
        ENDIF
        RST     2
        ENDM

INCLB   MACRO   ?D
        IF      NOT NUL ?D
        MVI     A,?D
        ENDIF
        RST     3
        ENDM


; RAM CONSTANT - CHANGE FOR DIFFERENT RAM LOCATION
LORAM   EQU     1000H   ; START OF RAM STORAGE


; NON-ZERO STATUS MEANS LINE BUFFER ADDRESS IS IN HL REG.
; ZERO STATUS MEANS NO BUFFER IS READY
NEXTIN  MACRO
        RST     4
        ENDM


; 8255 PARALLEL I/O EQUATES

PORTA   EQU     8               ; PORT A INPUT AND OUTPUT
PORTB   EQU     9               ; PORT B INPUT AND OUTPUT
PORTC   EQU     0AH             ; PORT C INPUT AND OUTPUT
CONTROL EQU     0BH             ; CONTROL PORT OUTPUT ONLY
; BAUD RATE EQUATES
BAUD384 EQU     4               ; DIVISOR FOR 38,400 BAUD
BAUD192 EQU     8               ; DIVISOR FOR 19,200 BAUD
BAUD96  EQU     16              ; DIVISOR FOR 9600 BAUD
BAUD48  EQU     32              ; DIVISOR FOR 4800 BAUD
BAUD24  EQU     64              ; DIVISOR FOR 2400 BAUD
BAUD12  EQU     128             ; DIVISOR FOR 1200 BAUD
BAUD600 EQU     256             ; DIVISOR FOR 600 BAUD
BAUD300 EQU     512             ; DIVISOR FOR 300 BAUD
BAUD150 EQU     1024            ; DIVISOR FOR 150 BAUD
BAUD134 EQU     1142            ; DIVISOR FOR 134.5 BAUD
BAUD110 EQU     1395            ; DIVISOR FOR 110 BAUD
BAUD75  EQU     2048            ; DIVISOR FOR 75 BAUD
BAUD50  EQU     3072            ; DIVISOR FOR 50 BAUD


; 8250 SERIAL I/O EQUATES

; REGISTER EQUATES
RBR     EQU     0               ; RECEIVE BUFFER REGISTER  (R)
THR     EQU     0               ; TRANSMIT HOLDING REGISTER (W)
IER     EQU     1               ; INTERRUPT ENABLE REGISTER  (W)
IIR     EQU     2               ; INTERRUPT IDENTIFICATION REGISTER (R)
LCR     EQU     3               ; LINE CONTROL REGISTER (R/W)
MCR     EQU     4               ; MODEM CONTROL REGISTER (R/W)
LSR     EQU     5               ; LINE STATUS REGISTER (R/W)
MSR     EQU     6               ; MODEM STATUS REGISTER (R/W)
DLL     EQU     0               ; DRIVER LATCH (LSB) (W)
DLM     EQU     1               ; DRIVER LATCH (MSB) (W)

; INTERRUPT ENABLE EQUATES
ERBFI   EQU     1               ; ENABLE RECEIVED DATA AVAILABLE INTERRUPT
ETBEI   EQU     2               ; ENABLE TRANSMITTER HOLDING REGISTER EMPTY
ELSI    EQU     4               ; ENABLE RECEIVER LINE STATUS INTERRUPT
EDSSI   EQU     8               ; ENABLE MODEM STATUS INTERRUPT

; INTERRUPT IDENTIFICATION EQUATES
IPEND   EQU     1               ; '0' IF INTERRUPT PENDING
IID0    EQU     2               ; INTERRUPT IDENTIFICATION BIT 0
IID1    EQU     4               ; INTERRUPT IDENTIFICATION BIT 1

; LINE CONTROL EQUATES
WLS0    EQU     1               ; WORD LENGTH SELECT BIT 0
WLS1    EQU     2               ; WORD LENGTH SELECT BIT 1
STB     EQU     4               ; STOP BIT SELECT
PEN     EQU     8               ; PARITY ENABLE
EPS     EQU     10H             ; EVEN PARITY SELECT
SPTY    EQU     20H             ; STICK PARITY
SBRK    EQU     40H             ; SET BREAK
DLAB    EQU     80H             ; DRIVER LATCH ACCESS BIT

; MODEM CONTROL EQUATES
DTR     EQU     1               ; DATA TERMINAL READY
RTS     EQU     2               ; REQUEST TO SEND
OUT1    EQU     4               ; OUT1 LINE ON 8250
OUT2    EQU     8               ; OUT2 LINE ON 8250
LOOP    EQU     10H             ; MODEM LOOP CONTROL BIT

; LINE STATUS EQUATES
DR      EQU     1               ; DATA READY
OE      EQU     2               ; OVERRUN ERROR
PE      EQU     4               ; PARITY ERROR
FE      EQU     8               ; FRAMING ERROR
BI      EQU     10H             ; BREAK INTERRUPT
THRE    EQU     20H             ; TRANSMITTER HOLDING REGISTER EMPTY
TSRE    EQU     40H             ; TRANSMITTER SHIFT REGISTER EMPTY

; MODEM STATUS EQUATES
DCTS    EQU     1               ; DATA CLEAR TO SEND
DDSR    EQU     2               ; DELTA DATA SET READY
TERI    EQU     4               ; TRAILING EDGE RING INDICATOR
DRLSD   EQU     8               ; DELTA RECEIVE LINE SIGNAL DETECT
CTS     EQU     10H             ; CLEAR TO SEND
DSR     EQU     20H             ; DATA SET READY
RI      EQU     40H             ; RING INDICATE
RLSD    EQU     80H             ; RECEIVE LINE SIGNAL DETECT

RIMD    EQU     17H             ; REQUEST INITIALIZATION MODE CONTROL BYTE
```

```
; CHARACTER FORMAT EQUATES FOR STANDARD EQUIPMENT
MOD15   EQU     STB     ; FOR MODEL 15 BAUDOT TTY
ASR33   EQU     WLS1+PEN+STB    ; FOR MODEL ASR33 TTY
HORST   EQU     WLS1+WLSO       ; FOR HORST'S SYSTEM
BOB     EQU     WLS1+PEN        ; FOR BOB'S TERMINAL
RICHARD EQU     WLS1+PEN+EPS    ; FOR RICHARD'S TERMINAL
APPLE   EQU     WLS1+WLSO+STB   ; FOR APPLE COMPUTER

MSE     EQU     08H     ; MASK SET ENABLE BIT


; COMMON COMMUNICATIONS AREA

; CIRCULAR TERMINAL BUFFER VARIABLES

CCA     EQU     LORAM   ; ADDRESS OF BEGINNING OF COMMON COMMUNICATIONS AREA
CTBIE   EQU     CCA+4   ; CURRENT TERMINAL BUFFER INPUT ENTRY
OTBE    EQU     CCA+6   ; OLDEST TERMINAL BUFFER ENTRY
TBIP    EQU     CCA+8   ; TERMINAL BUFFER INPUT POINTER
TBOP    EQU     CCA+0AH ; TERMINAL BUFFER OUTPUT POINTER
LTBOE   EQU     CCA+0CH ; LAST TERMINAL BUFFER OUTPUT ENTRY
CTBOE   EQU     CCA+0EH ; CURRENT TERMINAL BUFFER OUTPUT ENTRY

; CIRCULAR LINE BUFFER VARIABLES

LBPE    EQU     CCA+12H ; LINE BUFFER PROCESSING ENTRY
CLBE    EQU     CCA+14H ; CURRENT LINE BUFFER ENTRY ADDRESS
OLBE    EQU     CCA+16H ; OLDEST LINE BUFFER ENTRY
LBIP    EQU     CCA+18H ; LINE BUFFER INPUT POINTER
LBOP    EQU     CCA+1AH ; LINE BUFFER OUTPUT POINTER

; MISCELLANEOUS


STAT1   EQU     CCA     ; MAINLINE STATUS BYTE

TBOFLO  EQU     CCA+3   ; TERMINAL BUFFER OVERFLOW STATUS
BUFCOUNT EQU    CCA+1CH ; CURRENT INPUT BUFFER COUNT
OUTCOUNT EQU    CCA+1DH ; CURRENT BUFFER OUTPUT BYTES REMAINING
WAIT    EQU     CCA+40H ; CHARACTER DELAY VALUE
;
XMITSYNC        EQU     CCA+070H        ;SYNC AND UNDERWAY FLAG
TXUND   EQU     01H     ;TIP TRANSMIT UNDERWAY
TXSYNC  EQU     02H     ;TIP MAY TRANSMIT 1 CHARACTER

CR      EQU     0DH     ; ASCII CARRIAGE RETURN
LF      EQU     0AH     ; ASCII LINE FEED

MODE    EQU     CCA+31H ; MODE OF OPERATION
CONNECTED       EQU     80H
DISCONNECTING   EQU     40H
CONNECTING      EQU     20H
ACCEPTCON       EQU     01H     ;ACCEPT CONNECT REQUEST BIT

TRUE    EQU     0FFH    ; FOR IF CONDITION TESTS
FALSE   EQU     0       ; FOR IF CONDITION TESTS
; ********************************************************************
; **            CONFIGURATION EQUATES                               **
; **      VALUES CHANGE FOR EVERY CONFIGURATION                     **
; ********************************************************************


FORMAT  EQU     HORST   ; CURRENT CHARACTER FORMAT
BAUDRAT EQU     BAUD12  ; CURRENT BAUD RATE

CTSFLOW EQU     TRUE    ; IF FLOW CONTROL FROM DIGITAL EQUIPMENT TO TIP
                ; USING THE CLEAR TO SEND EIA LINE IS IMPLEMENTED.  THE
                ; DIGITAL EQUIPMENT SHOULD STOP SENDING DATA WHEN CTC DROPS.
                ; MUTUALLY EXCLUSIVE WITH XONFLOW.  (ALWAYS TRUE AT PRESENT)
```

```
XONFLOW EQU     FALSE   ; FOR FLOW CONTROL FROM DIGITAL EQUIPMENT TO TIP
                ; USING XON-XOFF PROTOCOL.  THE DIGITAL EQUIPMENT SHOULD
                ; STOP SENDING DATA UPON RECEIVING CONTROL-S (DC3) AND
                ; RESUME SENDING UPON RECEIVING CONTROL-Q (DC1).
                ; MUTUALLY EXCLUSIVE WITH CTSFLOW. (NOT IMPLEMENTED YET)

CUSHION EQU     10      ; THE NUMBER OF BYTES THAT MAY STILL BE SENT AFTER
                ; FLOW CONTROL ACTS TO STOP TRANSFER OF DATA FROM THE
                ; DIGITAL EQUIPMENT TO THE TIP.  NOTE THAT SOME EQUIPMENT
                ; ONLY BREAKS AT END OF LINE.

        ORG     0C00H   ; WHERE THIS PROGRAM'S EPROM STARTS

        PAGE
; ENTRY JUMP TABLE

        JMP     TIPINIT ; INITIALIZATION ENTRY POINT CALLED BY LIP
        JMP     RST55   ; INTERRUPT FROM 8250
        JMP     $       ; UNUSED INTERRUPT ENTRY POINT
        JMP     DISPATCH        ; TO DISPATCHER ROUTINE
RIMBUF  DB      12,RIMD,'VE3PKT'        ; CONNECTION BUFFER
TERMNO  DB      0B9H    ; THIS NODES TERMINAL NUMBER

TIPINIT:
        ; SET BAUD RATE IN SERIAL PORT
        MVI     A,DLAB
        OUT     LCR
        MVI     A,LOW BAUDRAT
        OUT     DLL             ; BAUD RATE DIVISOR LSB
        MVI     A,HIGH BAUDRAT
        OUT     DLM             ; BAUD RATE DIVISOR MSB
        ; DEFINE CHARACTER FORMAT OF SERIAL DATA
        MVI     A,FORMAT
        OUT     LCR     ; UPDATE LINE CONTROL REGISTER
        ; UNMASK INTERRUPTS FROM SERIAL INTERFACE
        RIM             ; GET CURRENT INTERRUPT MASK IN A
        ANI     00000110B       ; RESET RST5.5 MASK BIT
        ORI     MSE     ; SET MASK SET ENABLE BIT
        SIM             ; ENABLE RST5.5 INTERRUPTS

        ; CLEAR OUT RECEIVE BUFFER REGISTER
        IN      RBR

        ; ENABLE RECEIVED DATA AVAILABLE AND MODEM INTERRUPTS
        MVI     A,ERBFI+EDSSI
        OUT     IER     ; UPDATE INTERRUPT ENABLE REGISTER

        ; BRING UP RLSD AND CLEAR TO SEND FOR TERMINAL
        ; RTS = CTS, OUT1 = RLSD
        MVI     A,OUT1+RTS
        OUT     MCR     ; UPDATE MODEM CONTROL REGISTER

        ; RETURN TO LIP FOR COMPLETION OF INITIALIZATION
        RET
        PAGE
RST55:  PUSH    PSW
        PUSH    H
        PUSH    D
        PUSH    B
        IN      IIR     ; GET INTERRUPT IDENTIFICATION INFORMATION
        CPI     IID1    ; IS IT RECEIVED DATA AVAILABLE INTERRUPT?
        JZ      RXINT   ; YES, GO TO RECEIVE INTERRUPT ROUTINE
        ORA     A       ; MODEM INTERRUPT?
        JNZ     EXIT    ; UNKNOWN INTERRUPT, RETURN
;
; MODEM INTERRUPT.  SEE IF CONNECT STATUS CHANGE (VIA DSR)
; OR IF DATA SYNCHRONIZATION (VIA CTS)
```

```
;
          IN      MSR      ;GET MODEM STATUS
          MOV     C,A      ;SAVE IT
          ANI     DDSR     ;DELTA DSR?
          JZ      MSINT2   ;NO. HOW ABOUT DELTA CTS?
          MOV     A,C      ;YES. GET THE DSR STATE
          ANI     DSR      ;ISOLATE BIT
          LDA     MODE     ;PICK UP LINK STATUS BIT
          JZ      MSINT1   ;DSR LOW. FORCE A DISCONNECT
          ORI     ACCEPTCON        ;HIGH. ALLOW CONNECTIONS
          STA     MODE     ;TELL THE LIP
          JMP     EXIT
MSINT1:   ANI     OFFH-ACCEPTCON  ;CLEAR THE ACCEPT CONNECTION BIT
          STA     MODE
          JMP     FORCEDISC        ;FORCE A DISCONNECT
MSINT2:   MOV     A,C      ;GET STATUS AGAIN
          ANI     DCTS     ;IS IT CTS CHANGE?
          JZ      EXIT     ;NO. EXIT
          MOV     A,C      ;GET NEW CTS STATUS
          ANI     CTS
          JZ      EXIT     ;NOW DOWN. DISPATCH
          LDA     XMITSYNC         ;NOW UP. SET THE SYNC BIT
          ORI     TXSYNC   ;TO ENABLE 1 BYTE TO HOST
          STA     XMITSYNC
          JMP     EXIT     ;AND DISPATCH
RXINT:    IN      RBR      ; READ DATA FROM SERIAL PORT
;         ANI     7FH      ; TURN OFF HIGH ORDER BIT
          CPI     18H      ; CONTROL X
          JNZ     TESTDIS
          LDA     MODE     ;GET LINK STATE
          ANI     CONNECTING+CONNECTED     ;ANY ONE CONNECTED TO US
          JNZ     EXIT     ;YES. IGNORE THE REQUEST
          MVI     A,0      ; 0 FOR CONNECT
          RST     6
          JMP     EXIT
TESTDIS:
          CPI     19H      ; CONTROL Y
          JNZ     OFLOTEST
FORCEDISC:
          LDA     MODE     ;GET LINK STATE
          ANI     CONNECTING+CONNECTED     ;ANY ONE CONNECTED TO US
          JZ      EXIT     ;NO. IGNORE THE REQUEST
          MVI     A,1      ; 1 FOR DISCONNECT
          RST     6
          JMP     EXIT
OFLOTEST:
          MOV     C,A      ; SAVE DATA BYTE IN C
          LDA     TBOFLO   ; GET OVERFLOW INDICATOR
          ORA     A        ; IS THE TERMINAL BUFFER FULL?
          JNZ     EXIT     ; YES. DON'T DO ANYTHING
          LHLD    OTBE
          XCHG
          LHLD    TBIP
          INCTB   1        ; HL = TBIP+1
          CZ      OVERFLOW          ; INDICATE OVERFLOW IF BUFFER FULL
          JZ      EXIT     ; EXIT INTERRUPT ROUTINE IF OVERFLOW
          SHLD    TBIP
          MOV     M,C      ; PUT DATA IN BUFFER
          LXI     H,0
          SHLD    WAIT     ; RESET CHARACTER DELAY COUNT
          LXI     H,BUFCOUNT
          INR     M        ; INCREMENT BUFFER DATA COUNT
          MOV     A,C      ; GET DATA BYTE IN A
          CPI     LF       ; IS IT A LINE FEED?
          CZ      CLOSE    ; YES. GO TO CLOSE THIS ENTRY
          LDA     BUFCOUNT          ; GET DATA COUNT
          CPI     250      ; IS IT 250 BYTES OR MORE?
          CNC     CLOSE    ; YES. GO TO CLOSE ENTRY
          LDA     TBOFLO
          ORA     A        ; IS TERMINAL BUFFER FULL NOW?
```

```
          JNZ     EXIT     ; YES. NOTHING MORE TO DO
          LHLD    OTBE
          XCHG
          LHLD    TBIP
          INCTB   CUSHION  ; IS BUFFER CUSHION FREE?
          JNC     EXIT     ; YES. EXIT
;
;         NO. CHARACTER IN CUSHION. DROP CTS AT HOST TO STOP SENDING
;
          LDA     MODE     ;GET LINK STATE
          ANI     CONNECTING+CONNECTED     ;ANY ONE CONNECTED TO US
          MVI     A,OUT1   ;RLSD LINE FOR HOST BUT NO CTS
          JZ      FLOW1    ;NO-ONE CONNECTED. LEAVE DTR LINE LOW
          ORI     DTR      ;SOMEONE THERE. RAISE (OR LEAVE HIGH) DTR LINE
FLOW1:    OUT     MCR      ; TURN OFF CLEAR TO SEND
          JMP     EXIT     ; EXIT THE INTERRUPT ROUTINE

EXIT:     POP     B
          POP     D
          POP     H
          POP     PSW
          EI
          RET

DISPATCH:
          LXI     B,XMITSYNC       ;GET SYNC WORD ADDRESS
          LDA     MODE     ;FETCH TNC MODE
          ANI     CONNECTING+CONNECTED
          JZ      DISPATCHD ;NO.
          MVI     A,DTR+OUT1+RTS   ;CONNECTED. BRING UP DTR FOR HOST
          JMP     DISPATCHX
DISPATCHD:
          DI
          LDAX    B        ;ALLOW ALL DATA TO GO OUT IF
          ORI     TXSYNC   ;NOT CONNECTED IN ORDER TO
          STAX    B        ;KEEP TIP BUFFERS DRAINED
          EI
          MVI     A,OUT1+RTS       ;DROP DTR FOR HOST
DISPATCHX:
          OUT     MCR
          IN      LSR      ; GET LINK STATUS REGISTER
          ANI     TSRE     ; IS THE TRANSMIT SHIFT REGISTER EMPTY?
          JZ      DELAYCHECK       ; NO. THEN GET OUT OF HERE
          LDAX    B        ;GET SYNC WORD
          MOV     D,A      ;PRESERVE FOR LATER TEST
          ANI     TXSYNC   ;CAN WE SEND A BYTE?
          JZ      DELAYCHECK       ;NO. SKIP
          MOV     A,D
          ANI     TXUND    ;TRANSMIT ALREADY UNDERWAY?
          JNZ     TXCONT   ;YES. CONTINUE WITH TRANSMISSION
;
          NEXTIN           ;NO. TRY TO GET ANOTHER BUFFER
          JZ      DELAYCHECK       ;NONE AVAILABLE
          MOV     A,M      ; GET DATA LENGTH IN BUFFER HEADER
          STA     OUTCOUNT         ; AND SAVE IT FOR INTERRUPT ROUTINE
          INCLB   3
          SHLD    LBOP     ; LBOP = OLBE + 3
          DI
          LDAX    B        ;GET SYNC WORD
          ORI     TXUND    ;SET UNDERWAY
          STAX    B
          EI
TXCONT:   LHLD    LBOP
          INCLB   1
          SHLD    LBOP     ; LBOP = LBOP+1
          MOV     A,M
          DI
          OUT     THR      ; OUTPUT DATA AT LBOP
          LDAX    B
```

```
        ANI     OFFH-TXSYNC     ;KILL THE SYNC BIT
        STAX    B
        EI
        LXI     H,OUTCOUNT
        DCR     M
        JNZ     DELAYCHECK
        DI
        LDAX    B       ;GET SYNC WORD
        ANI     OFFH-TXUND      ;KILL TRANSMIT UNDERWAY
        STAX    B
        EI
;
;       CHECK IF IT IS TIME TO FLUSH THE TIP BUFFER TO THE LIP.
;       IF NO CHARACTER RECEIVED WITHIN THE LAST 17 MS (BY GSB CALC)
;
DELAYCHECK:
        LDA     TBOFLO
        ORA     A
        RNZ
        IN      MCR     ; GET MODEM CONTROL REGISTER
        ANI     RTS     ; IS CLEAR TO SEND UP?
        JNZ     DELAY2  ; YES. SKIP CTS TURN ON TEST
        LHLD    OTBE
        XCHG
        LHLD    TBIP
        INCTB   CUSHION ; IS BUFFER CUSHION FREE?
        RC              ; NO. DON'T TURN ON CTS OR DO DELAY CHECK
        LDA     MODE    ;GET LINK STATE
        ANI     CONNECTING+CONNECTED    ;ANY ONE CONNECTED TO US
        MVI     A,RTS+OUT1
        JZ      FLOW2   ;NO-ONE CONNECTED. LEAVE DTR LINE LOW
        ORI     DTR     ;SOMEONE THERE. RAISE (OR LEAVE HIGH) DTR LINE
FLOW2:  OUT     MCR     ; TURN ON CLEAR TO SEND
DELAY2:
        LDA     BUFCOUNT        ; GET CURRENT DATA COUNT
        CPI     0       ; IS COUNT ZERO?
        RZ              ; YES. NOTHING TO CHECK. RETURN
        LHLD    WAIT
        INX     H
        SHLD    WAIT
        MOV     A,H
        CPI     1H
        RNZ
        CALL    DISABLERX       ; DISABLE RECEIVER INTERRUPTS
        LDA     BUFCOUNT
        CPI     0
        JZ      ENABLERX
        LDA     TBOFLO
        ORA     A
        JNZ     ENABLERX
        CALL    CLOSE   ; CLOSE OFF THIS ENTRY
        JMP     ENABLERX        ; ENABLE RECEIVER INTERRUPTS AND EXIT

; SUBROUTINE TO CLOSE OFF TERMINAL BUFFER ENTRY AND PASS
; TO CONTROL OF LIP.
CLOSE:
        LXI     H,BUFCOUNT
        MOV     A,M
        MVI     M,0     ; BUFCOUNT = 0
        LHLD    CTBIE
        MOV     M,A     ; SET UP DATA COUNT IN HEADER
        LHLD    OTBE
        XCHG
        LHLD    TBIP
        INCTB   1
        SHLD    CTBIE
        CZ      OVERFLOW
        INCTB   1
        SHLD    TBIP
        CZ      OVERFLOW
        RET     ; RETURN TO CALLER

; SUBROUTINE TO DISABLE 8250 RECEIVE INTERRUPTS
DISABLERX:
        DI
        IN      IER
        ANI     OFFH-ERBFI
        OUT     IER
        EI
        RET

; SUBROUTINE TO ENABLE 8250 RECEIVE INTERRUPTS
ENABLERX:
        DI
        IN      IER
        ORI     ERBFI
        OUT     IER
        EI
        RET

; SUBROUTINE TO TURN ON TERMINAL BUFFER OVERFLOW INDICATOR
OVERFLOW:
        MVI     A,OFFH
        STA     TBOFLO
        RET

        END
```

□

"WISH LIST"

Plans are being made in several parts of the country to create a new version of the TNC board. Perhaps it will have additional **capabilities, such as two HDLC ports,** so that it can be used as **a linking repeater.**

Your comments and suggestions, possibly based on experience, **are solicited.**

Examples:
- **"autobaud" TIP software to perform** automatic speed setting
- **combine LIP Terminal-to-terminal** and Station-to-terminal **programs.**
- **X-on, X-off flow control support in the TIP.**
- **make system configuration** commands transparent to the data **flow. e.g. Connect-disconnect commands.**
- **on TNC board, supply test power** to pins 9 and 10 of DB25 **modem connector.**

□

2

```
        TITLE   'VADCG TNC MODULE - LIP-TT (LAST REVISED 0725 01-SEP-81)'
;*********************************************************************
;**     VADCG TERMINAL NODE COMMUNICATIONS PROGRAM  -  MODULE LIP   **
;**        BY DOUG LOCKHART, VE7APU            DECEMBER, 1979       **
;*********************************************************************
; LAST CHANGED: JUNE 30, 1980
; CHANGED 20-AUG-81. FIXES THRU MEMO 10 AND SPECIAL RABBS CODE
;  WILL ALLOW CONNECTIONS TO BE MADE IF TIP HAS SET THE 'ACCEPTCON'
;  BIT IN 'MODE'. CONNECT AND DISCONNECT MESSAGES CHANGED SLIGHTLY
;
; THIS PROGRAM IS DESIGNED TO RUN IN THE VADCG TERMINAL NODE CONTROLLER
; TOGETHER WITH A TERMINAL INTERFACE PROGRAM. THIS PROGRAM DRIVES AN
; INTEL 8273 HDLC/SDLC PROTOCOL CONTROL CHIP USING INTERRUPTS AND RUNS
; IN ROM MEMORY.

        MACLIB  LIB85   ; GET 8085 MACRO LIBRARY

INCTB   MACRO   ?D
        MVI     A,?D
        RST     2
        ENDM

INCLB   MACRO   ?D
        MVI     A,?D
        RST     3
        ENDM

; MEMORY CONFIGURATION EQUATES
LORAM   EQU     01000H  ; FIRST BYTE OF CONTIGUOUS RAM AREA
HIRAM   EQU     01FE0H  ; LAST BYTE OF CONTIGUOUS RAM AREA

; 8273 PORT EQUATES

STAT73  EQU     10H     ; STATUS REGISTER
COMM73  EQU     10H     ; COMMAND REGISTER
PARM73  EQU     11H     ; PARAMETER REGISTER
RESL73  EQU     11H     ; RESULT REGISTER
TXIR73  EQU     12H     ; TRANSMIT INTERRUPT RESULT REGISTER
RXIR73  EQU     13H     ; RECEIVER INTERRUPT RESULT REGISTER
TXDATA  EQU     18H     ; OUTPUT DATA PORT
RXDATA  EQU     20H     ; INPUT DATA PORT

; 8273 STATUS REGISTER BIT EQUATES

CPBF    EQU     20H     ; COMMAND PARAMETER BUFFER FULL BIT
CRBF    EQU     10H     ; COMMAND RESULT BUFFER FULL BIT
CBF     EQU     40H     ; COMMAND BUFFER FULL BIT
CBSY    EQU     80H     ; COMMAND BUSY BIT
TXIRA   EQU     01H     ; TRANSMIT INTERRUPT RESULT AVAILABLE
RXIRA   EQU     02H     ; RECEIVE INTERRUPT RESULT AVAILABLE
TINT    EQU     04H     ; TX INTERRUPT BIT IN STATUS REGISTER
RINT    EQU     08H     ; RX INTERRUPT BIT IN STATUS REGISTER

; 8273 PORT A BIT EQUATES
CD      EQU     02H     ; CARRIER DETECT BIT

RIMD    EQU     17H     ; REQUEST INITIALIZATION
DISC    EQU     53H     ; DISCONNECT WITH POLL BIT SET

DISCACK EQU     43H     ; DISCONNECT ACKNOWLEDGEMENT
RIMACK  EQU     07H     ; REQUEST FOR INITIALIZATION ACKNOWLEDGEMENT
RR      EQU     01H     ; RECEIVE READY
RNR     EQU     05H     ; RECEIVE NOT READY
```

```
; TERMINAL INTERFACE PROGRAM EQUATES
TIP     EQU     0C00H   ; ADDRESS OF TERMINAL INTERFACE PROGRAM EPROM
TIPINIT EQU     TIP     ; TIP INITIALIZATION ENTRY POINT
TIPINT1 EQU     TIP+3   ; INTERRUPT ENTRY POINT #1 IN TIP
TIPINT2 EQU     TIP+6   ; INTERRUPT ENTRY POINT #2 IN TIP
TIPENT  EQU     TIP+9   ; ADDRESS OF DISPATCHER ENTRY POINT IN TIP
RIMBUF  EQU     TIP+0CH ; ADDRESS OF INITIALIZATION FRAME IN TIP
TERMNO  EQU     TIP+14H ; ADDRESS OF TERMINAL NUMBER

        PAGE
; COMMON COMMUNICATIONS AREA

; CIRCULAR TERMINAL BUFFER VARIABLES

CCA     EQU     LORAM   ; ADDRESS OF BEGINNING OF RAM AREA
TBOFLO  EQU     CCA+3   ; TERMINAL BUFFER OVERFLOW STATUS BYTE
CTBIE   EQU     CCA+4   ; CURRENT TERMINAL BUFFER INPUT ENTRY
OTBE    EQU     CCA+6   ; OLDEST TERMINAL BUFFER ENTRY
TBIP    EQU     CCA+8   ; TERMINAL BUFFER INPUT POINTER
TBOP    EQU     CCA+0AH ; TERMINAL BUFFER OUTPUT POINTER
LTBOE   EQU     CCA+0CH ; LAST TERMINAL BUFFER OUTPUT ENTRY
CTBOE   EQU     CCA+0EH ; CURRENT TERMINAL BUFFER OUTPUT ENTRY

; CIRCULAR LINE BUFFER VARIABLES

LBPE    EQU     CCA+12H ; LINE BUFFER PROCESSING ENTRY
CLBE    EQU     CCA+14H ; CURRENT LINE BUFFER ENTRY ADDRESS
OLBE    EQU     CCA+16H ; OLDEST LINE BUFFER ENTRY
LBIP    EQU     CCA+18H ; LINE BUFFER INPUT POINTER
LBOP    EQU     CCA+1AH ; LINE BUFFER OUTPUT POINTER

; MISCELLANEOUS

NS      EQU     CCA+10H ; NUMBER SENT
NR      EQU     CCA+11H ; NUMBER RECEIVED

T1      EQU     5000    ; LINE TIMEOUT DELAY VALUE

BUFCOUNT EQU    CCA+1CH ; CURRENT INPUT BUFFER COUNT FOR TIP
OUTCOUNT EQU    CCA+1DH ; CURRENT OUTPUT BUFFER COUNT FOR TIP

STAT1   EQU     CCA     ; MAINLINE STATUS BYTE
; STAT1 BYTE BIT EQUATES
DLY     EQU     80H     ; CHANNEL CLEAR TIMEOUT IN PROGRESS
TMOUT   EQU     40H     ; LINE TIMEOUT IN PROGRESS
RNRSA   EQU     20H     ; STATION HAS ACKNOWLEDGED RNR
POLLR   EQU     10H     ; STATION DEMANDS RESPONSE
RIMR    EQU     08H     ; REQUEST FOR INITIALIZATION RECEIVED
RNRR    EQU     04H     ; RECEIVE NOT READY HAS BEEN RECEIVED
DISCR   EQU     02H     ; DISCONNECT FRAME HAS BEEN RECEIVED
RNRS    EQU     01H     ; RECEIVE NOT READY HAS BEEN SEND

STAT2   EQU     CCA+1   ; LINK STATUS
; STAT2 BYTE BIT EQUATES
RXBUSY  EQU     80H     ; RECEIVER IS BUSY
TXBUSY  EQU     40H     ; TRANSMITTER IS BUSY

FBIT    EQU     10H     ; FINAL BIT HAS BEEN SENT

STAT3   EQU     CCA+2   ; LINE BUFFER OVERFLOW STATUS WHEN NE 0
; STAT3 BYTE BIT EQUATES
OFLO    EQU     RNRS+RNRSA      ; OVERFLOW STATUS BITS USED TO EXCLUSIVE
                                ; OR WITH RNR SENT & ACKNOWLEDGED BITS

USBUFFER EQU    CCA+1EH ; U AND S-FRAME TRANSMIT BUFFER (2 BYTES)
USCMD   EQU     USBUFFER+1      ; POINTS TO CONTROL FIELD IN USBUFFER

; RECEIVER COMMAND BUFFER
RCMDBUF EQU     CCA+20H ; RECEIVE COMMAND BUFFER
```

```
RCMDLEN EQU     CCA+22H ; RECEIVE BUFFER LENGTH
RCMDA1  EQU     CCA+24H ; SELECTIVE RECEIVE ADDRESS 1
RCMDA2  EQU     CCA+25H ; SELECTIVE RECEIVE ADDRESS 2

; TRANSMITTER COMMAND BUFFER
TCMDBUF EQU     CCA+26H ; TRANSMIT COMMAND BUFFER
TCMDLEN EQU     CCA+28H ; TRANSMIT BUFFER LENGTH
TCMDADR EQU     CCA+2AH ; TRANSMIT ADDRESS FIELD
TCMDCON EQU     CCA+2BH ; TRANSMIT CONTROL FIELD

TIME    EQU     CCA+2CH ; TIMER COUNT (2 BYTES)
RANVAL  EQU     CCA+2EH ; RANDOM VALUE (1 BYTE)
HASBUF  EQU     CCA+2FH ; TERMINAL INTERFACE HAS BUFFER FOR OUTPUT
                        ; WHEN = 'FF' - USED BY NEXTIN ROUTINE
TOCOUNT EQU     CCA+30H ; NUMBER OF LINE TIMEOUTS
;
MODE    EQU     CCA+31H ; MODE OF OPERATION
CONNECTED       EQU     80H
DISCONNECTING   EQU     40H
CONNECTING      EQU     20H
ACCEPTCON       EQU     01H     ;TIP WILL ALLOW CONNECTIONS

CBUF    EQU     CCA+32H ; CONNECT/DISCONNECT BUFFER
CBUFLEN EQU     CCA+32H ; DATA LENGTH
CBUFCMD EQU     CCA+33H ; COMMAND
CBUFNAM EQU     CCA+34H ; OWN CALL
CBUFCON EQU     CCA+3AH ; CALL OF CONNECTION
;CBUFEND        EQU     CCA+40H ; END OF CONNECT BUFFER        (MEMO 8)

STACK   EQU     CCA+70H ; TOP OF STACK AREA
XMITSYNC        EQU     CCA+70H ;SYNCHRONIZING AND TX UNDERWAY FLAG IN TIP

LBA     EQU     STACK+1 ; FIRST BYTE OF LINE BUFFER AREA
LBAEND  EQU     (HIRAM + LBA) / 2       ; NEXT BYTE AFTER END OF LBA
LBALEN  EQU     (LBAEND - LBA)  ; LENGTH OF LINE BUFFER AREA
TBA     EQU     LBAEND  ; FIRST BYTE OF TERMINAL BUFFER AREA
TBAEND  EQU     HIRAM   ; NEXT BYTE AFTER END OF TBA
TBALEN  EQU     (TBAEND - TBA)  ; LENGTH OF TERMINAL BUFFER AREA

        PAGE
; NODE COMMUNICATION PROGRAM MAINLINE
; INTERRUPT VECTORS

        ORG     0400H   ;WE GET HERE FROM ROM RESIDENT DLLOAD
RESTART: JMP    INIT    ; ENTRY POINT WHEN RESTART BUTTON IS PRESSED
        ORG     0408H
RST1:   JMP     $
        ORG     0410H
RST2:   PUSH    B
        PUSH    D
        CALL    INCRTB
        POP     D
        POP     B
        RET
        ORG     0418H
RST3:   PUSH    B
        PUSH    D
        CALL    INCRLB
        POP     D
        POP     B
        RET
        ORG     0420H
RST4:   PUSH    D
        JMP     NEXTIN2 ; GETS NEXT LINE BUFFER ENTRY FOR TIP
        ORG     0428H
RST5:   PUSH    B       ; SAVE BC ON STACK
        JMP     COMPHLDE        ; GO TO COMPARE HL WITH DE
```

```
        ORG     042CH
RST55:  JMP     TIPINT1 ; GO TO TERMINAL INTERFACE PROGRAM JUMP TABLE
        ORG     0430H
RST6:   JMP     ALTMODE
        ORG     0434H
RST65:  JMP     TXINT   ; 8273 TRANSMIT INTERRUPT
        ORG     0438H
RST7:   JMP     TIPINT2 ; GO TO TERMINAL INTERFACE PROGRAM JUMP TABLE
        ORG     043CH
RST75:  JMP     RXINT   ; 8273 RECEIVE INTERRUPT
        PAGE
NEXTIN: LHLD    LBPE
        XCHG            ; DE <-- LINE BUFFER PROCESSING ENTRY
        LHLD    OLBE    ; HL <-- OLDEST LINE BUFFER ENTRY
        RST     5       ; OLBE = LBPE?
;       RZ              ; RETURN, ZERO STATUS IF NOTHING TO PROCESS (MEMO 10)
        JNZ     NEXTIN0 ;YES, GO TO PROCESS IT                     (MEMO 10)
        LDA     STAT3   ;GET OVERFLOW STATUS IN A                  (MEMO 10)
        RRC             ;GET OVERFLOW BIT INTO CARRY               (MEMO 10)
        RNC             ;BUFFER EMPTY, RETURN WITH ZERO STATUS     (MEMO 10)
NEXTIN0:                ;EXACTLY FULL IF WE FALL THRU TO HERE      (MEMO 10)
        LDA     HASBUF  ; GET BUFFER OWNERSHIP INICATOR
        ORA     A       ; DOES TIP HAVE BUFFER
        JZ      NEXTIN3 ; NO GO TO CHECK FOR VALID BUFFER
        MVI     A,0
        STA     HASBUF  ; INDICATE TIP HAS NO BUFFER
NEXTIN1:
        MOV     A,M     ; GET LENGTH OF DATA
        ADI     4       ; ADD HEADER LENGTH
        RST     3       ; POINT TO NEXT ENTRY
        SHLD    OLBE    ; UPDATE OLDEST ENTRY POINTER
        MVI     A,0     ; TURN OFF OVERFLOW INDICATION
        STA     STAT3   ; UPDATE LINK STATUS BYTE
        JMP     NEXTIN  ; CHECK NEXT ENTRY
NEXTIN3:
        INCLB   2       ; POINT AT ADDRESS FIELD
        MVI     A,0
        CMP     M       ; SKIP THIS ENTRY? (A-FIELD = 0?)
        LHLD    OLBE    ; GET OLDEST ENTRY BACK IN HL
        JZ      NEXTIN1 ; YES, SKIP IT
        MVI     A,0FFH
        STA     HASBUF  ; INDICATE TIP HAS A BUFFER
        RET             ; RETURN WITH NON-ZERO STATUS
                        ; HL HAS ADDRESS OF ENTRY TO BE PROCESSED BY TIP
NEXTIN2:
        CALL    NEXTIN
        POP     D
        RET

ALTMODE:
        CPI     1       ; DISCONNECT?
        JZ      ALT4    ; YES
ALT1:   LHLD    CTBIE   ; COME HERE FOR CONNECT
;       INCTB   2       ;POINT TO FIRST BYTE OF CALL FROM         (MEMO 9)
        INCTB   1       ;POINT TO ONE BYTE BEFORE DATA            (MEMO 9)
        SHLD    TBIP    ;SET POINTER TO RE-WRITE BUFFER ENTRY     (MEMO 9)
        INCTB   1       ;POINT TO CALL SIGN TO CONNECT TO         (MEMO 9)
        MVI     A,0                                               (MEMO 9)
        STA     BUFCOUNT ;INDICATE EMPTY ENTRY                    (MEMO 9)
        LXI     D,CBUFCON       ; POINT TO FIRST BYTE OF CALL TO
        MVI     B,6
ALT2:   MOV     A,M
        STAX    D
        INX     D
        INCTB   1
        DCR     B
        JNZ     ALT2
        LDA     MODE
        ANI     ACCEPTCON
```

```
         ORI     CONNECTING
ALT3:    STA     MODE
         LDA     STAT1
         ANI     OFFH-RNRS-RNRSA-RNRR
         STA     STAT1
         MVI     A,0
         STA     TOCOUNT
         RET
ALT4:    LDA     MODE
         ANI     ACCEPTCON
         ORI     DISCONNECTING
         JMP     ALT3

; RESTART 5 ROUTINE TO COMPARE HL WITH DE
; RETURNS ZERO CONDITION IF HL = DE
; RETURNS CARRY CONDITION IF HL > DE
; THIS ROUTINE ONLY AFFECTS STATUS. IT DOES NOT AFFECT REGISTERS
COMPHLDE:
         MOV     B,A      ; SAVE ACCUMULATOR. BC ALREADY SAVED ON STACK
         MOV     A,D
         CMP     H
         JNZ     COMPRET  ; EXIT IF NOT EQUAL
         MOV     A,E
         CMP     L        ; COMPARE L WITH E
COMPRET:
         MOV     A,B      ; RESTORE ACCUMULATOR
         POP     B        ; RESTORE BC FROM STACK
         RET              ; RETURN TO CALLER
         PAGE
; INITIALIZATION CODE
; ENTERED FROM RESTART INTERRUPT
INIT:    DI               ; DISABLE INTERRUPTS
         LXI     D,HIRAM  ; HIGHEST ADDRESS TO CLEAR
         LXI     H,LORAM  ; LOWEST ADDRESS TO CLEAR
CLEAR:
         MVI     M,0
         INX     H
         MOV     A,D
         CMP     H
         JNZ     CLEAR          ; NOT OVER YET
         MOV     A,E            ; GET LOW ORDER BYTE
         CMP     L              ; IS IT THE ONE?
         JNZ     CLEAR          ; NO, KEEP CLEARING
; SET UP STACK POINTER
         LXI     SP,STACK

; INITIALIZE LINE BUFFER VARIABLES
         LXI     H,LBA
         SHLD    CLBE     ; CURRENT LINE BUFFER ENTRY
         SHLD    OLBE     ; OLDEST LINE BUFFER ENTRY
         SHLD    LBPE     ; LINE BUFFER PROCESSING ENTRY
         LXI     H,LBA+3  ; POINT TO END OF HEADER AREA
         SHLD    LBIP     ; LINE BUFFER INPUT POINTER
; INTIALIZE TERMINAL BUFFER VARIABLES
         LXI     H,TBA    ; POINT TO START OF TERMINAL BUFFER AREA
         SHLD    OTBE     ;OLDEST TERMINAL BUFFER ENTRY
         SHLD    CTBIE    ; SET UP CURRENT TERMINAL BUFFER INPUT ENTRY ADDRESS
         INX     H        ; INCREMENT ADDRESS BY ONE
                          ; TO POINT TO END OF HEADER
         SHLD    TBIP     ; TERMINAL BUFFER INPUT POINTER
; INITIALIZE CONNECT BUFFER
         LXI     H,CBUF
         LXI     D,RIMBUF
         MVI     B,8      ; B<-- LENGTH OF INITIALIZATION DATA
MOVLP:   LDAX    D
         MOV     M,A
```

```
         INX     H
         INX     D
         DCR     B
         JNZ     MOVLP
; INITIALIZE TRANSMIT COMMAND BUFFER
         LXI     H,0C804H        ; TRANSMIT FRAME COMMAND
         SHLD    TCMDBUF
         LDA     TERMNO
         STA     TCMDADR ; SET UP THIS TERMINAL'S ADDRESS

; INITIALIZE RECEIVE COMMAND BUFFER
         LXI     H,250
         SHLD    RCMDLEN
         LXI     H,0C002H     ;GENERAL RECEIVE COMMAND
         SHLD    RCMDBUF

; INITIALIZE 8273 MODE REGISTERS
         LXI     H,DATTRANS
         CALL    CMDOUT    ; SET DATA TRANSFER MODE REGISTER
         LXI     H,OPMODE
         CALL    CMDOUT    ; SET OPERATING MODE REGISTER
         LXI     H,SERIALIO
         CALL    CMDOUT    ; SET SERIAL I/O MODE REGISTER
         LXI     H,DTR
         CALL    CMDOUT    ; TURN ON DATA TERMINAL READY LINE
         MVI     A,19H     ; UNMASK RST6.5 AND RST7.5 INTERRUPTS
         SIM               ; SET INTERRUPT MASK

; CALL INITIALIZATION ENTRY POINT IN TERMINAL INTERFACE PROGRAM
         CALL    TIPINIT ; LET TIP INITIALIZE ITSELF

         EI                ; ENABLE INTERRUPTS
         PAGE
DISPATCH:
         LDA     STAT2     ; GET LINK STATUS BYTE IN A
         ANI     TXBUSY    ; IS TRANSMITTER ACTIVE?
         CZ      XMIT      ; IF NOT, SEE IF TRANSMITTER SHOULD BE ACTIVATED
         LDA     STAT2     ; GET LINKSTATUS BYTE IN A
         ANI     TXBUSY    ; IS TRANSMITTER ACTIVE
         CZ      RBUSYTEST      ; IF NOT, TEST RECEIVER STATUS
         LHLD    CLBE
         XCHG              ; DE <-- CURRENT LINE BUFFER ENTRY
         LHLD    LBPE      ; HL <-- LINE BUFFER PROCESSING ENTRY
         RST     5         ; IS CURRENT ENTRY SAME AS PROCESSED ENTRY?
         CNZ     INPROC    ; IF NOT, GO TO PROCESS RECEIVED FRAMES
         CALL    TIPENT    ; ALLOW TERMINAL INTERFACE PROGRAM TO RUN
         JMP     DISPATCH ; LOOP LOOKING FOR SOMETHING TO DO

; 8273 INITIALIZATION COMMAND BUFFERS
DATTRANS: DB     1,97H,1 ; INTERRUPT ON DATA TRANSFERS
OPMODE:  DB      1,91H,0FH      ; NO HDLC ABORT, NO EOP INTERRUPTS
; EARLY TRANSMIT INTERRUPT, BUFFERED MODE, PREFRAME SYNC, FLAG STREAM
SERIALIO: DB     1,0A0H,1       ; NRZI MODE
DTR:     DB      1,0A3H,4       ; ENABLE DATA TERMINAL READY
         PAGE
; ON ENTRY HL CONTAINS LBPE, DE CONTAINS CLBE
INPROC:
         LDA     MODE
         ANI     CONNECTED
         JNZ     INFRAME ; YES, PROCESS RECEIVED FRAME
         INCLB   3       ; POINT HL AT CONTROL FIELD
         MVI     A,1
         ANA     M
         JZ      INCRLBPE
         MVI     A,0EFH
         ANA     M
         CPI     RIMACK
         JZ      CMPCALL
         CPI     DISCACK
         JNZ     SKIP
CMPCALL:
         INCLB   7      ; POINT HL AT CALL IN INPUT BUFFER
         LXI     D.CBUFNAM       ; POINT DE AT MY CALL
         MVI     B.6
         LDA     MODE   ;GET TIP STATUS
         ANI     ACCEPTCON      ;ACCEPTING CONNECTIONS?
         JZ      INCRLBPE       ;NO. TREAT AS IF CALL DIDNT MATCH
CMPLOOP:                        ;YES. CHECK CALL IN BUFFER
         LDAX    D      ; GET RIMBUF DATA BYTE
         CMP     M      ; IS IT SAME AS BUFFER?
         JNZ     INCRLBPE
         INX     D      ; COME HERE TO TEST NEXT POSITION
         INCLB   1
         DCR     B      ; DECREMENT COUNT
         JNZ     CMPLOOP ; NOT FINISHED. CONTINUE COMPARISON
INPROC1:
         LHLD    LBPE
         INCLB   3
         MVI     A.0EFH
         ANA     M
         CPI     RIMACK  ; IS IT RIM?
         JZ      CONNECT
         MOV     A.M
         ANI     POLLR
         JZ      DISC2
         LDA     STAT1
         ORI     DISCR
         STA     STAT1   ; INDICATE DISCONNECT RECEIVED
DISC2:   LDA     MODE
         ANI     ACCEPTCON
         STA     MODE    ; INDICATE MONITOR MODE BUT PRESERVE TIP FLAG
SHOWD:   INCLB   7       ; POINT TO MY CALL IN BUFFER
         MVI     B.5
MOVED:   MVI     M.'D'
         INCLB   1
         DCR     B
         JNZ     MOVED
         MVI     M.0DH   ;STUFF A CR FOR RABBS
         INCLB   1
         CALL    STOPRX
         LXI     H.0C002H
         SHLD    RCMDBUF ; GENERAL RECEIVE COMMAND
         JMP     INCRLBPE

STOPRX:  LXI     H.DISRX
         CALL    CMDOUT
         LDA     STAT2
         ANI     OFFH-RXBUSY
         STA     STAT2
         RET

CONNECT:
         MOV     A.M     ; GET C-FIELD
         ANI     10H     ; IS POLL BIT ON?
         LDA     STAT1
         JZ      CNCT2
         ORI     RIMR
CNCT2:   ANI     OFFH-TMOUT
         STA     STAT1
         CALL    STOPRX
         LHLD    LBPE
         INCLB   4       ; POINT HL AT CONNECTION CALL SIGN
         LXI     D.CBUFCON
         MVI     B.6
CONMOVE:
         MOV     A.M
         STAX    D
         INX     D
```

```
         INCLB   1
         DCR     B
         JNZ     CONMOVE
         LHLD    LBPE
         INCLB   2
         MOV     A.M
         STA     RCMDA1
         STA     RCMDA2
         LXI     H.0C104H
         SHLD    RCMDBUF ; INITIALIZE FOR SELECTIVE RECEIVE
         LHLD    LBPE
         INCLB   10      ; POINT TO THIS TERMINAL'S CALL IN BUFFER
         MVI     B.5
MOVEC:   MVI     M.'C'
         INCLB   1
         DCR     B
         JNZ     MOVEC
         MVI     M.0DH   ;STUFF A CR FOR RABBS
         INCLB   1
         LXI     H.0
         SHLD    NS      ; NS=NR=0
         MVI     A.0
         STA     TOCOUNT
         LDA     MODE
         ANI     ACCEPTCON
         ORI     CONNECTED
         STA     MODE
         CALL    INCRLBPE        ; POINT TO NEXT ENTRY
         SHLD    CLBE
         RET

DISCON:  LDA     STAT1   ; GET MAINLINE STATUS BYTE
         ORI     DISCR   ; SET DISCONNECT RECEIVED BIT
         STA     STAT1   ; UPDATE MAINLINE STATUS BYTE
SKIP:    LHLD    LBPE    ; HL <-- LINE BUFFER PROCESSING ENTRY
         INCLB   2       ; POINT HL AT A-FIELD
         MVI     M.0     ; A-FIELD=0 (SKIP THIS ENTRY)
INCRLBPE:
         LHLD    LBPE    ; HL <-- LINE BUFFER PROCESSING ENTRY
         MOV     A.M     ; A <-- LENGTH OF DATA
         ADI     4       ; A <-- DATA LENGTH + HEADER LENGTH
         RST     3       ; HL <-- POINTER TO NEXT ENTRY
         SHLD    LBPE    ; UPDATE LINE BUFFER PROCESSING ENTRY
         RET             ; RETURN TO DISPATCHER

; ON ENTRY HL HAS LBPE. DE HAS CLBE.
INFRAME:
         INCLB   3       ; POINT HL AT C-FIELD
         MOV     A.M     ; A <-- C-FIELD
         ANI     1       ; IS IT AN INFORMATION FRAME?
         JZ      IFRAME  ; YES. GO TO HANDLE I-FRAMES
         LDA     STAT1   ; GET MAINLINE STATUS BYTE
         ANI     OFFH-TMOUT      ; TMOUT = 0
         STA     STAT1   ; UPDATE MAINLINE STATUS BYTE
         MOV     A.M     ; A <-- C-FIELD
         ANI     0AH     ; IS IT RR OR RNR?
         JNZ     BADFRAMEY       ; NO. NOT A VALID U OR S FRAME
         MOV     A.M     ; A <-- C-FIELD
         ANI     14H     ; SELECT POLL AND RNR BITS
         MOV     B.A     ; SAVE IN B
         LDA     STAT1   ; A <-- MAINLINE STATUS BYTE
         ANI     OFFH-POLLR-RNRR ; POLLR=0 AND RNRR=0
         ORA     B       ; OR ON REQUIRED BITS
         MOV     B.A     ;SAVE STAT 1 IN B                        (MEMO 2)
         ANI     POLLR   ;IS POLL BIT ON                          (MEMO 2)
         MOV     A.B     ;GET STAT1 BACK                          (MEMO 2)
         JNZ     INF2    ;YES. DON'T CHANGE RNRSA                 (MEMO 2)
         ANI     RNRS    ;TEST RNRS                               (MEMO 2)
         MOV     A.B     ;GET STAT1 AGAIN                         (MEMO 2)
```

```
        JZ      INF1    ;GO SET RNRSA TO 0 IF RNRS IS 0         (MEMO 2)
        ORI     RNRSA   ; RNRSA = RNRS= 1                       (MEMO 2)
        JMP     INF2    ;SKIP NEXT LINE                         (MEMO 2)
INF1:   ANI     OFFH-RNRSA      ;RNRSA = RNRS = 0               (MEMO 2)
INF2:                   ;                                       (MEMO 2)
        STA     STAT1   ; UPDATE MAINLINE STATUS BYTE
        CALL    NRPROC  ; PROCESS NR
        JMP     SKIP    ; SKIP THIS ENTRY

BADFRAMEY:
        MOV     B,M     ; SAVE C-FIELD
        MOV     A,M
        ANI     OFFH-POLLR
        CPI     DISCACK
        JZ      BADFRAME
        CPI     RIMACK
        JNZ     BADFRAME
        LHLD    NS
        MOV     A,H
        ORA     L
        JNZ     BADFRAME
        MOV     A,B     ; RECOVER C-FIELD
        ANI     POLLR
        JZ      SKIP
        JMP     INPROC1
BADFRAMEX:
        POP     B       ; DISCARD RETURN ADDRESS
BADFRAME:
        LDA     MODE
        ANI     ACCEPTCON
        STA     MODE
        STA     TOCOUNT
        LDA     STAT1
        ANI     OFFH-RNRS-RNRSA-RNRR-TMOUT
        ORI     DISCR
        STA     STAT1
        JMP     SHOWD

; ENTRY WITH HL POINTING AT C-FIELD
NRPROC: MOV     A,M     ; A <-- C-FIELD
        ANI     OEOH    ; SELECT NR BITS
        RRC             ; SHIFT RIGHT FOUR BITS TO GET
        RRC             ; NR RECEIVED IN POSITION FOR COMPARISON
        RRC
        RRC
        MOV     B,A     ; SAVE NR IN B
NRPROC1:
        LDA     NS      ; A <-- NS
        CMP     B       ; IS NR RECEIVED = NS?
        RZ              ; YES, GOOD RETURN
        LHLD    LTBOE
        XCHG            ; DE <-- LTBOE
        LHLD    OTBE    ; HL <-- OTBE
        RST     5       ; IS OTBE = LTBOE?
        JZ      BADFRAMEX       ; YES, INCONGRUOUS NR RECEIVED
        LDA     NS      ; A <-- NS
        ADI     2       ; NS <-- NS + 1
        ANI     OEH     ; CLEAR OUT ANY OVERFLOW
        STA     NS      ; UPDATE NS
        MOV     A,M     ; A <-- LENGTH OF DATA IN ENTRY
        ADI     2       ; A <-- DATA LENGTH + HEADER LENGTH
        RST     2       ; HL <-- POINTER TO NEXT ENTRY
        SHLD    OTBE    ; UPDATE OLDEST ENTRY POINTER
        MVI     A,O
        STA     TBOFLO  ; INDICATE BUFFER SPACE AVAILABLE FOR TIP
        JMP     NRPROC1 ; CONTINUE PROCESSING NR RECEIVED

; ENTRY WITH HL POINTING AT C-FIELD
IFRAME: MOV     A,M     ; A <-- C-FIELD
```

```
        ANI     POLLR   ; TEST FOR POLL BIT
        LDA     STAT1   ; GET MAINLINE STATUS BYTE IN A
        JZ      SHORTTO ; START SHORT LINE TIMEOUT IF NO POLL BIT
        ANI     OFFH-TMOUT      ; TMOUT = 0
        JMP     IFRAME2 ; GO TO UPDATE MAINLINE STATUS
SHORTTO:
        LXI     H,200   ; SHORT TIMEOUT COUNT - DURATION SHOULD BE
                        ; EQUIVALENT TO A 256 BYTE TRANSMISSION
        SHLD    TIME
        ANI     OFFH-DLY        ; DLY = 0
        ORI     TMOUT   ; TMOUT=1
IFRAME2:
        ANI     OFFH-RNRS-RNRSA ; RNRS = 0, RNRSA = 0
        ORI     POLLR   ; POLLR = 1
        STA     STAT1   ; UPDATE MAINLINE STATUS
        LHLD    LBPE
        MOV     A,M     ; A <-- LENGTH OF DATA
        CPI     0       ; IS DATA LENGTH = 0?
        JZ      BADFRAME        ; YES BAD PROTOCOL
        INCLB   3       ; HL POINTS TO C-FIELD
        CALL    NRPROC  ; PROCESS NR RECEIVED
        LHLD    LBPE    ; HL <-- LBPE
        INCLB   3       ; POINT HL AT C-FIELD
        MOV     A,M     ; A <-- C-FIELD
        ANI     OEH     ; SELECT NS BITS
        RLC             ; SHIFT LEFT FOUR BITS
        RLC             ; FOR COMPARISON TEST
        RLC
        RLC
        MOV     B,A     ; SAVE IN B
        LDA     NR      ; A <-- NR
        CMP     B       ; IS NR = NS RECEIVED?
        JNZ     SKIP    ; NO, ENTRY OUT OF SEQUENCE, SKIP IT
        ADI     20H     ; NR = NR + 1 (COME HERE IF NS IS OK)
        STA     NR      ; UPDATE NR
        JMP     INCRLBPE        ; GO TO POINT TO NEXT ENTRY
        PAGE
RBUSYTEST:
        LDA     STAT2   ; GET LINK STATUS IN A
        ANI     RXBUSY  ; IS THE RECEIVER OPERATING?
        JNZ     RXACTIVE        ; YES
        LHLD    OLBE
        XCHG            ; DE <-- OLBE
        LHLD    CLBE    ; HL <-- CLBE
        RST     5
        JNZ     RBUSYTEST1
        LDA     STAT3   ; GET OVERFLOW STATUS BYTE
        RRC             ; TEST FOR LINE BUFFER OVERFLOW
        RC              ; RETURN, NO ROOM IN BUFFER AT ALL
        INCLB   3
        JMP     RBUSYTEST2      ; SKIP TESTS BECAUSE BUFFER IS EMPTY
RBUSYTEST1:
        INCLB   3       ; CLBE <- CLBE + 3
        RZ              ; RETURN IF CLBE+3 = OLBE
        RC              ; RETURN IF CLBE + 3 > OLBE
RBUSYTEST2:
        SHLD    LBIP    ; LBIP = CLBE + 3
        CALL    RXCMDOUT ; ************************
        MVI     A,RXBUSY        ; RXBUSY = 1
        STA     STAT2   ; UPDATE LINK STATUS BYTE
        RET             ; RETURN TO DISPATCHER
RXACTIVE:
        LDA     STAT1   ; GET MAINLINE STATUS BYTE TO TEST
        ANI     TMOUT   ; IS LINE TIMEOUT IN PROGESS?
        RZ              ; NO NEED TO PROCESS TIMER
        CALL    TESTCH  ; TEST IF CHANNEL BUSY
        RNZ             ; YES IT IS, NO NEED TO PROCESS TIMER
TIMER:  LHLD    TIME    ; GET TIMEOUT COUNT IN HL
        DCX     H       ; DECREMENT COUNT
        SHLD    TIME    ; UPDATE TIMER VALUE
        MOV     A,H
        ORA     L       ; IS COUNT = 0?
        RNZ             ; NO, RETURN
        LDA     STAT1   ; COME HERE IF TIMEOUT EXPIRED
        ANI     OFFH-TMOUT-DLY  ; INDICATE TIMER NOT RUNNING
        STA     STAT1   ; UPDATE MAINLINE STATUS BYTE
        XRA     A       ; SETS ZERO STATUS TO INDICATE EXPIRATION
        RET             ; RETURN TO DISPATCHER
        PAGE
TXINT:  PUSH    PSW     ; SAVE PSW ON STACK
        PUSH    H
        IN      STAT73  ; GET 8273 STATUS BYTE
        RRC             ; GET RESULT INDICATOR INTO CARRY
        LHLD    TBOP    ; POINT HL AT OUTPUT POINT
        JC      TXRESULT        ; YES, GO HANDLE RESULT
        INX     H
        MVI     A,LOW TBAEND
        CMP     L
        JNZ     TBOPUPDATE
        MVI     A,HIGH TBAEND
        CMP     H
        JNZ     TBOPUPDATE
        LXI     H,TBA
TBOPUPDATE:
        SHLD    TBOP
        MOV     A,M
        OUT     TXDATA
        POP     H
        POP     PSW
        EI
        RET

TXRESULT:
        PUSH    D
        PUSH    B
        IN      TXIR73  ; READ TRANSMIT RESULT
        CPI     OCH     ; IS IT EARLY INTERRUPT RESULT?
        JZ      TXEARLY ; YES, GO HANDLE EARLY INTERRUPT
        CPI     ODH     ; IS IT NORMAL COMPLETION RESULT?
        JNZ     TXBUF   ; NO, RETRANSMIT FRAME
;       LDA     STAT2   ; GET LINK STATUS BYTE                  (MEMO 7)
;       ANI     OFFH-TXBUSY     ; INDICATE TRANSMITTER NOT BUSY (MEMO 7)
;       STA     STAT2   ; UPDATE LINK STATUS BYTE               (MEMO 7)
        XRA     A       ;                                       (MEMO 7)
        STA     STAT2   ;INDICATE BOTH TX AND RX NOT BUSY       (MEMO 7)
        LDA     TCMDCON ;GET LAST CONTROL FIELD SENT            (MEMO 7)
        ANI     FBIT    ; HAS FINAL BIT JUST BEEN SENT?
        JZ      EXIT
        LXI     H,T1    ; GET LINE TIMEOUT VALUE INTO HL
        SHLD    TIME    ; SET TIMER UP FOR LINE TIMEOUT
        LDA     STAT1   ; GET MAINLINE STATUS BYTE
        ORI     TMOUT   ; INDICATE LINE TIMEOUT IN PROGRESS
        STA     STAT1   ; UPDATE MAINLINE STATUS INICATORS
        JMP     EXIT    ; EXIT FROM INTERRUPT ROUTINE

TXEARLY:
;       LXI     D,USBUFFER+2                            (MEMO 7)
;       MVI     A,1                                     (MEMO 7)
;       RST     2       ; HL <- TBOP+1                  (MEMO 7)
;       SHLD    CTBOE   ; CTBOE <- TBOP+1               (MEMO 7)
;       JZ      EXIT    ; EXIT IF WE TRANSMITTED U OR S-FRAME   (MEMO 7)
;       XCHG                                            (MEMO 7)
;       LHLD    LTBOE                                   (MEMO 7)
;       RST     5       ; IS CTBOE = LTBOE?             (MEMO 7)
;       JNZ     TXBUF   ; NO, GO TO TRANSMIT BUFFER AT CTBOE    (MEMO 7)
;       LDA     STAT2   ; GET LINK STATUS BYTE          (MEMO 7)
;       ANI     FBIT    ; FINAL BIT BEEN SENT?          (MEMO 7)
;       JNZ     EXIT    ; IF YES, THEN EXIT INTERRUPT ROUTINE   (MEMO 7)
;       LXI     H,USBUFFER                              (MEMO 7)
;       SHLD    CTBOE   ; POINT TO U OR S-FRAME BUFFER  (MEMO 7)
```

```
        LDA     TCMDCON ;GET LAST CONTROL FIELD SENT            (MEMO 7)
        ANI     FBIT+RR ;TEST CONTROL FIELD BITS                (MEMO 7)
        JNZ     EXIT    ;EXIT IF POLL BIT ON OR NOT I-FRAME     (MEMO 7)
        INCTB   1       ;POINT TO START OF NEXT ENTRY           (MEMO 7)
        SHLD    CTBOE   ;UPDATE CURRENT ENTRY POINTER           (MEMO 7)
        XCHG            ;                                       (MEMO 7)
        LHLD    LTBOE   ;                                       (MEMO 7)
        RST     5       ;WAS THAT THE LAST ENTRY TO SEND?       (MEMO 7)
        JNZ     TXBUF   ;NO, GO AND SEND THIS ONE               (MEMO 7)
        LXI     H,USBUFFER ;                                    (MEMO 7)
        SHLD    CTBOE   ;UPDATE TO SEND USBUFFER                (MEMO 7)
TXBUF:  CALL    TXFRAME ; TRANSMIT BUFFER AT CTBOE
        JMP     EXIT    ; EXIT FROM INTERRUPT ROUTINE
        PAGE
RXINT:  PUSH    PSW     ; SAVE PSW ON STACK
        PUSH    H
        IN      STAT73  ; GET 8273 STATUS BYTE
        ANI     RXIRA   ; IS INTERRUPT RESULT AVAILABLE?
        JNZ     RXRESULT
        LHLD    LBIP
        INX     H
        MVI     A,LOW LBAEND
        CMP     L
        JNZ     OFLOTEST
        MVI     A,HIGH LBAEND
        CMP     H
        JNZ     OFLOTEST
        LXI     H,LBA
OFLOTEST:
        LDA     OLBE    ; GET LOW BYTE
        CMP     L
        JNZ     INLINE
        LDA     OLBE+1
        CMP     H
        JZ      OVERFLOW
INLINE:
        IN      RXDATA
        MOV     M,A
        SHLD    LBIP
        POP     H
        POP     PSW
        EI
        RET

RXRESULT:
        PUSH    D
        PUSH    B
        IN      RXIR73
        MOV     C,A
        LHLD    CLBE    ; POINT HL AT CURRENT HEADER
RXRES:  IN      STAT73  ; READ 8273 STATUS
        ANI     RINT    ; RECEIVE INTERRUPT STILL PENDING?
        JZ      CHKRIC  ; NO, GO CHECK RIC
        IN      STAT73  ; READ 8273 STATUS AGAIN
        ANI     RXIRA   ; IS INTERRUPT RESULT AVAILABLE?
        JZ      RXRES   ; NO, CONTINUE TESTING
        IN      RXIR73  ; GET RESULT IN A
        MOV     M,A     ; PUT RESULT IN HEADER
        INX     H
        MVI     A,LOW LBAEND
        CMP     L
        JNZ     RXRES
        MVI     A,HIGH LBAEND
        CMP     H
        JNZ     RXRES
        LXI     H,LBA
        JMP     RXRES   ; GO TO CHECK IF MORE RESULTS
CHKRIC: MOV     A,C     ; GET RIC IN A
```

```
        CPI     OEOH    ; GOOD RETURN?
        JC      BADSTAT ; NO. GO HANDLE BAD RETURN CODE
        LHLD    LBIP    ; HL <-- LBIP
        INCLB   1       ; HL <-- LBIP + 1
        SHLD    CLBE    ; CLBE <-- LBIP + 1
        LHLD    OLBE
        XCHG            ; DE <-- OLBE
        LHLD    LBIP    ; HL <-- LBIP
        INCLB   4       ; HL <-- LBIP + 4
        JZ      OVERFLOW1       ;OOPS, NO ROOOM LEFT            (MEMO 1)
        JC      OVERFLOW1       ; OOPS                          (MEMO 1)
        JMP     STARTRX ; GOOD. GO AND START RECEIVER           (MEMO 1)
OVERFLOW:
        IN      RXDATA  ;READ DATA TO CLEAR INTERRUPT           (MEMO 3)
        PUSH    D       ;SAVE REST OF REGISTERS                 (MEMO 1)
        PUSH    B       ;                                       (MEMO 1)
OVERFLOW1:              ;                                       (MEMO 1)
        LXI     H,DISRX ; POINT TO DISABLE RECEIVER COMMAND
        CALL    CMDOUT  ; DISABLE RECEIVER
        LDA     STAT2   ; GET LINK STATUS BYTE INTO A
        ANI     OFFH-RXBUSY     ; INDICATE RECEIVER NOT BUSY
        STA     STAT2   ; UPDATE LINK STATUS BYTE
        MVI     A,OFLO  ; INDICATE LINE BUFFER HAS OVERFLOWED
        STA     STAT3   ; UPDATE LINK STATUS BYTE
        JMP     EXIT    ; EXIT FROM INTERRUPT ROUTINE
BADSTAT:
        LHLD    CLBE    ; HL <-- CURRENT ENTRY POINTER
        INCLB   3       ; HL <-- CLBE + 3
STARTRX:
        SHLD    LBIP    ; UPDATE LINE BUFFER INPUT POINTER
        CALL    RXCMDOUT        ; ISSUE RECEIVE COMMAND TO 8273
EXIT:   POP     B       ; RESTORE REGISTERS
        POP     D
        POP     H
        POP     PSW     ; RESTORE PSW
        EI              ; ENABLE INTERRUPTS
        RET             ; RETURN TO INTERRUPTED CODE

DISRX   DW      OC500H  ; DISABLE RECEIVER COMMAND BUFFER
        PAGE
XMIT:   LDA     STAT1   ; GET MAINLINE STATUS
        ANI     DLY+TMOUT       ; TEST FOR TIMER ACTIVITY
        JM      DELAY   ; HANDLE CHANNEL AVAILABLE DELAY
        RNZ             ; IF RECEIVE TIMEOUT IN PROGRESS. RETURN TO MAINLINE
        LDA     STAT1   ; GET MAINLINE STATUS IN A
        MOV     C,A     ; SAVE STATUS IN C
        ANI     RIMR+DISCR+POLLR        ; MUST WE TRANSMIT?
        JNZ     CDTEST  ; YES. GO TO TEST IF CHANNEL IS FREE
        LDA     MODE
        ANI     DISCONNECTING+CONNECTING
        JNZ     CDTEST
        LDA     STAT3   ; GET OVERFLOW STATUS BYTE IN A
        XRA     C       ; EXCLUSIVE OR WITH MAINLINE STATUS BYTE
        ANI     OFLO    ; IF ZERO. THEN RNRS=RNRSA=OFLO
        JNZ     CDTEST
        MOV     A,C     ; GET MAINLINE STATUS AGAIN
        ANI     RNRR    ; CAN STATION RECEIVE I-FRAMES?
        RNZ             ; NO. RETURN TO DISPATCHER
        LHLD    CTBIE
        XCHG
        LHLD    OTBE
        RST     5       ; IS CURRENT TERM. BUFFER = OLDEST TERM. BUFFER?
        RZ              ; YES. NOTHING TO TRANSMIT. RETURN TO DISPATCHER
CDTEST: CALL    TESTCH  ; TEST IF CHANNEL IS OCCUPIED
        RNZ             ; YES. RETURN IF CHANNEL IS OCCUPIED
        LDA     STAT1   ; GET MAINLINE STATUS IN A
        ORI     DLY     ; INDICATE CHANNEL CLEAR DELAY
        STA     STAT1   ; UPDATE MAINLINE STATUS BYTE
        CALL    RANDOM  ; SET TIMER COUNT TO RANDOM VALUE
        RET             ; RETURN TO MAINLINE
```

```
DELAY:  CALL TIMER      ; PROCESS DELAY TIMER
        RNZ             ; RETURN. CHANNEL CLEAR DELAY STILL ACTIVE
        CALL    TESTCH  ; TEST IF CHANNEL IS BUSY
        RNZ             ; YES IT IS. RETURN TO MAINLINE
        LXI     H,DISRX ; POINT TO DISABLE RECEIVER COMMAND
        CALL    CMDOUT  ; DISABLE RECEIVER
        MVI     A,TXBUSY        ; RXBUSY=0,TXBUSY=1,FBIT=0
        STA     STAT2   ; UPDATE LINK STATUS BYTE
        CALL    STARTTRANSMIT   ; SET UP FRAMES FOR TRANSMISSION
        LDA     STAT1   ; GET MAINLINE STATUS INTO A
        ANI     OFFH-POLLR      ; POLLR = 0
        STA     STAT1   ; UPDATE MAINLINE STATUS
        JMP     TXFRAME ; START TRANSMITTER AND RETURN TO DISPATCHER
        RET             ; RETURN TO DISPATCHER
        PAGE
STARTTRANSMIT:
        LDA     STAT1
        ANI     RIMR
        JNZ     SENDCNCTBUF
        LDA     MODE
        ANI     CONNECTED
        JZ      NOCONNECT
        LXI     H,USBUFFER
        SHLD    CTBOE   ; MAKE BUFFER FOR U & S-FRAMES CURRENT
        LDA     STAT1   ; GET MAINLINE STATUS BYTE IN A
        ANI     RNRR    ; HAS RNR BEEN RECEIVED?
        JNZ     ENDSETUP        ; YES. NOT READY FOR I-FRAMES
        LHLD    CTBIE
        XCHG            ; DE <-- CTBIE
        LHLD    OTBE    ; HL <-- OTBE
        RST     5       ; DOES OTBE = CTBIE?
        JZ      ENDSETUP        ; YES. NOT READY TO SEND I-FRAMES
        SHLD    CTBOE
        PUSH    H
        LHLD    NS
        MOV     C,L     ; C <-- CURRENT NS VALUE
        MOV     B,H     ; B<-- CURRENT NR VALUE
        POP     H       ; HL <-- OTBE. DE HAS CTBIE
CSETUP: PUSH    H
        PUSH    H       ; BUFFER ADDRESS ON STACK TWICE
        MVI     A,1
        RST     2       ; POINT HL AT CONTROL FIELD IN BUFFER
        MOV     A,B     ; SET UP NR
        ORA     C       ; SET UP NS
        MOV     M,A     ; SET UP I-TYPE CONTROL FIELD IN BUFFER
        POP     H       ;POINT HL AT LENGTH FIELD IN BUFFER
        MOV     A,M     ; GET LENGTH BYTE IN A
        ADI     2       ; ADD 2 TO GET PAST HEADER
        RST     2       ; POINT HL AT NEXT ENTRY
        SHLD    LTBOE   ; UPDATE LAST OUTPUT ENTRY
        JZ      ENDSETUP
        MOV     A,C
        ADI     2
        ANI     OEH
        MOV     C,A
        ADI     2
        ANI     OEH
        PUSH    H       ; STACK <-- BUFFER POINTER
        LXI     H,NS    ; POINT HL AT NS VALUE
        CMP     M
        POP     H
        JZ      ENDSETUP
        POP     PSW
        JMP     CSETUP

NOCONNECT:
        LHLD    CTBIE
        XCHG
        LHLD    OTBE
        RST     5
        JZ      SENDCNCTBUF
        SHLD    CTBOE
SETLOOP:
;       MOV     A,M      ; LENGTH BYTE IN A                        (MEMO 6)
;       ADI     2        ; ADD 2 TO GET PAST HEADER                (MEMO 6)
;       RST     2        ; POINT TO NEXT ENTRY                     (MEMO 6)
;       RST     5        ; IS IT AT CTBIE?                         (MEMO 6)
;       JNZ     SETLOOP                                            (MEMO 6)
;       SHLD    LTBOE                                              (MEMO 6)
;       SHLD    OTBE                                               (MEMO 6)
;       INCTB   1        ; POINT TO C FIELD                        (MEMO 6)
;       MVI     M,10H    ; PUT   FINAL BIT ON                      (MEMO 6)
        SHLD    LTBOE    ;SAVE CURRENT ENTRY IN LTBOE              (MEMO 6)
        MOV     C,M      ;SAVE LENGTH IN C                         (MEMO 6)
        INCTB   1        ;POINT TO CONTROL FIELD                   (MEMO 6)
        MVI     M,0      ;SET CONTROL FIELD TO 0                   (MEMO 6)
        MOV     A,C      ;GET LENGTH IN A                          (MEMO 6)
        ADI     1        ;ADD 1 TO GET PAST HEADER                 (MEMO 6)
        RST     2        ;POINT TO NEXT ENTRY                      (MEMO 6)
        JNZ     SETLOOP  ;GO DO ANOTHER IF NOT FINISHED            (MEMO 6)
        LHLD    LTBOE    ;FINISHED. GET PREVIOUS ENTRY BACK        (MEMO 6)
        INCTB   1        ;POINT TO CONTROL FIELD AGAIN             (MEMO 6)
        MVI     M,10H    ;PUT FINAL BIT ON                         (MEMO 6)
        XCHG             ;GET CTBIE IN HL                          (MEMO 6)
        SHLD    LTBOE    ;INDICATE END OF LAST FRAME TO TRANSMIT   (MEMO 6)
        SHLD    OTBE     ;UPDATE OLDEST FRAME POINTER              (MEMO 6)
        RET

SENDCNCTBUF:
        MVI     A,0
        STA     CBUFCMD
        LDA     STAT1
        ANI     DISCR
        LXI     H,CBUFCMD
        JZ      RIMRTEST
        MVI     M,DISCACK
RIMRTEST:
        LDA     STAT1
        ANI     RIMR
        JZ      DISCGTEST
        MVI     M,RIMACK
DISCGTEST:
        LDA     MODE
        ANI     DISCONNECTING
        JZ      CONGTEST
        MVI     M,DISC
CONGTEST:
        LDA     MODE
        ANI     CONNECTING
        JZ      SENDCNCT1
        MVI     M,RIMD
SENDCNCT1:
        LDA     STAT1
        ANI     OFFH-RIMR-DISCR
        STA     STAT1
        LXI     H,CBUF
        SHLD    CTBOE
;       LXI     H,CBUFEND                                          (MEMO 8)
;       SHLD    LTBOE                                              (MEMO 8)
        LDA     TOCOUNT
        CPI     10H
        JNZ     UPCOUNT
        LDA     MODE
        ANI     ACCEPTCON
        STA     MODE
        STA     TOCOUNT
        LXI     H,0C002H
        SHLD    RCMDBUF  ; GENERAL RECEIVE
        RET
```

```
UPCOUNT:
        INR     A
        STA     TOCOUNT
        RET
ENDSETUP:
        LHLD    CTBOE    ; HL <-- CTBOE
        LXI     D,USBUFFER      ; DE <-- USBUFFER ADDRESS
        LXI     B,USCMD  ; BC <-- POINTER TO CONTROL BYTE IN USBUFFER
        LDA     STAT3    ; GET OVERFLOW STATUS BYTE IN A FOR TESTING
        ANI     OFLO     ; TEST OVERFLOW BITS
        LDA     NR       ; A <-- NR
        JZ      SENDRR   ; NO OVERFLOW. GO TO SEND RR
SENDRNR:
        ORI     RNR      ; SET UP RNR IN CONTROL FIELD
        STAX    B        ; UPDATE CONTROL FIELD IN BUFFER
        LDA     STAT1    ; GET MAINLINE STATUS BYTE IN A
        ANI     RNRS+RNRSA      ; TEST RNR SENT AND ACKNOWLEDGED BITS
        CPI     RNRS+RNRSA      ; ARE THEY BOTH ON?
        JZ      INSYNC   ; YES. BOTH SIDES OF LINK ARE SYNCHRONIZED
        LDA     STAT1    ; GET MAINLINE STATUS BYTE FOR UPDATE
        ORI     RNRS     ; RNRS = 1
        ANI     OFFH-RNRSA      ; RNRSA = 0
NOTSYNC:
        STA     STAT1    ; UPDATE MAINLINE STATUS BYTE
;       LDAX    B        ; A <-- CONTROL FIELD IN USBUFFER         (MEMO 2)
;       ORI     FBIT     ; TURN ON FINAL BIT                       (MEMO 2)
;       STAX    B        ; UPDATE CONTROL FIELD IN USBUFFER        (MEMO 2)
        CALL    POLLCHECK1      ;DO FINAL BIT PROCESSING           (MEMO 2)
        RST     5        ; DOES CTBOE = USBUFFER?
        RZ               ; YES. RETURN TO XMIT ROUTINE
        POP     B        ; DISCARD LAST I-FRAME ADDRESS
        RET              ; RETURN TO XMIT ROUTINE
SENDRR: ORI     RR       ; SET UP RNR CONTROL FIELD
        STAX    B        ; UPDATE CONTROL FIELD IN USBUFFER
        LDA     STAT1    ; GET MAINLINE STATUS BYTE IN A
        ANI     RNRS+RNRSA      ; TEST RNR SENT AND ACKNOWLEDGED BITS
        JZ      INSYNC   ; BOTH ARE ZERO. LINK IS SYNCHRONIZED
        LDA     STAT1    ; GET MAINLINE STATUS BYTE FOR UPDATE
        ANI     OFFH-RNRS       ; RNRS = 0
        ORI     RNRSA    ; RNRSA = 1
        JMP     NOTSYNC
INSYNC: RST     5        ; IS CTBOE = USBUFFER?
        JZ      POLLCHECK       ; YES, GO TO CHECK FOR POLL
        POP     H        ; HL <-- ADDRESS OF LAST I-FRAME ENTRY
        MVI     A,1
        RST     2        ; HL POINTS TO CONTROL FIELD IN TB ENTRY
        MOV     A,M      ; A <-- CONTROL FIELD
        ORI     FBIT     ; TURN ON FINAL BIT IN CONTROL FIELD
        MOV     M,A      ; UPDATE CONTROL FIELD IN BUFFER
        RET              ; RETURN TO XMIT ROUTINE
POLLCHECK:
        LDA     STAT1    ; GET MAINLINE STATUS BYTE
POLLCHECK1:                                                        (MEMO 2)
        ANI     POLLR    ; HAS RESPONSE BEEN REQUESTED BY STATION?
        RNZ              ; NO. RETURN TO XMIT ROUTINE
        LDAX    B        ; A <-- CONTROL FIELD FROM USBUFFER
        ORI     FBIT     ; TURN ON FINAL BIT IN CONTROL FIELD
        STAX    B        ; UPDATE CONTROL FIELD IN USBUFFER
        RET              ; RETURN TO XMIT ROUTINE
        PAGE
; SUBROUTINES

INCRLB: LXI     B,-LBALEN
        PUSH    B        ; STORE LENGTH OF BUFFER ON STACK
        LXI     B,LBAEND
        JMP     INCR
INCRTB: LXI     B,-TBALEN
        PUSH    B        ; STORE LENGTH OF BUFFER ON STACK
        LXI     B,TBAEND
```

13

```
INCR:   PUSH    B       ; STORE BUFFER END ADDRESS ON STACK
        MOV     C,A
        MVI     B,0     ; BC <-- INCREMENT COUNT
        RST     5       ; COMPARE HL WITH DE
        JC      INCR3
        DAD     B       ; HL <-- INCREMENTED VALUE
        RST     5
        POP     D       ; Z
        POP     B       ; -L
        PUSH    PSW     ; SAVE CURRENT STATUS
        XCHG            ; HL <- Z, DE <- X
        RST     5       ; Z > X ?
        XCHG            ; HL <- X, DE <- Z
        JC      INCR1
        DAD     B       ; SUBTRACT BUFFER LENGTH
INCR1:  POP     PSW     ; RESTORE STATUS
        RET
INCR3:  DAD     B
        XCHG
        XTHL
        RST     5       ; COMPARE HL TO DE
        POP     H
        JC      INCR4
        XCHG
        POP     B       ; BC <-- (-L)
        DAD     B       ; HL <-- X + (-L)
        RST     5
        RET
INCR4:  CMC             ; SET CARRY = 1
        XCHG            ; HL <-- X
        POP     B       ; CLEAR (-L) FROM STACK
        RET


; SETS UP RANDOM COUNT BETWEEN 100H AND 4FFH IN 'TIME'
RANDOM: LDA     RANVAL
        ADI     187
        STA     RANVAL  ; UPDATE RANDOM VALUE BYTE
        RLC
        RLC
        LXI     H,TIME  ; POINT TO 2 BYTE TIME FIELD
        MOV     M,A     ; SET UP LOW ORDER BYTE OF RANDOM COUNT
        ANI     03H     ; SELECT TWO LOW ORDER RANDOM BITS
        ADI     01H     ; MAKE SURE VALUE IS GREATER THAN 100H
        INX     H       ; POINT TO HIGH ORDER BYTE
        MOV     M,A     ; SET UP HIGH ORDER BYTE OF RANDOM COUNT
        RET             ; RETURN TO CALLING ROUTINE

; TESTS IF CARRIER DETECT IS ACTIVE
; ZERO STATUS RETURNED IF INACTIVE, NON-ZERO IF CHANNEL IS BUSY
TESTCH: LXI     H,RDPORTA       ; POINT TO COMMAND TO READ PORT A
        CALL    CMDOUT  ; ISSUE COMMAND TO READ PORT A
        CALL    IMDRLT  ; GET IMMEDIATE RESULT IN A
        ANI     CD      ; TEST CARRIER DETECT BIT
        RET             ; RETURN TO CALLING ROUTINE
RDPORTA DB      0,22H   ; READ PORT A COMMAND BUFFER

; STARTS TRANSMIT OF BUFFER POINTED TO BY CTBE
TXFRAME:
        LHLD    CTBOE   ; HL <-- CURRENT TRANSMIT BUFFER OUTPUT ENTRY
; ENTRY POINT TO TRANSMIT BUFFER AT HL
TXF1:   MOV     A,M     ; GET LENGTH BYTE FROM CURRENT ENTRY
        STA     TCMDLEN ; SET UP LENGTH IN TRANSMIT COMMAND BUFFER
        MVI     A,1
        RST     2       ; POINT HL AT CONTROL BYTE
        SHLD    TBOP    ; SET UP TERMINAL BUFFER OUTPUT POINTER
        MOV     A,M     ; GET CONTROL BYTE FROM CURRENT ENTRY
        STA     TCMDCON ; SET UP CONTROL FIELD IN TRANSMIT COMMAND BUFFER
;       ANI     FBIT    ; TEST FOR FINAL BIT IN C-FIELD         (MEMO 7)
;       JZ      TXF2    ; NOT PRESENT, BYPASS STATUS UPDATE     (MEMO 7)
;       MVI     A,TXBUSY+FBIT   ; TXBUSY=1,RXBUSY=0,FBIT=1      (MEMO 7)
;       STA     STAT2   ; UPDATE MAINLINE STATUS                (MEMO 7)
;TXF2:                                                          (MEMO 7)
        LXI     H,TCMDBUF ; POINT HL AT TRANSMIT COMMAND BUFFER (MEMO 7)
        CALL    CMDOUT  ; ISSUE TRANSMIT COMMAND TO 8273
        RET             ; RETURN TO CALLING ROUTINE


; FUNCTION: COMMAND DISPATCHER
; INPUTS: HL - COMMAND BUFFER ADDRESS
; OUTPUTS: NONE
; CALLS: NONE
; DESTROYS: A,B,H,L,F/F'S
; DESCRIPTION: CMDOUT ISSUES THE COMMAND + PARAMETERS
; IN THE COMMAND BUFFER POINTED AT BY HL.
;
RXCMDOUT:       ; ENTRY POINT FOR RECEIVE COMMAND
        LXI     H,RCMDBUF       ; POINT HL AT RECEIVE COMMAND BUFFER
CMDOUT: MOV     B,M     ; FIRST ENTRY IS PARAMETER COUNT
        INX     H       ; POINT AT COMMAND BYTE
CMD1:   IN      STAT73  ; READ 8273 STATUS
        RLC             ; ROTATE CBSY INTO CARRY
        JC      CMD1    ; WAIT UNTIL CBSY=0
        MOV     A,M     ; MOVE COMMAND BYTE INTO A
        OUT     COMM73  ; PUT COMMAND IN COMMAND REGISTER
CMD2:   MOV     A,B     ; GET PARAMETER COUNT
        ANA     A       ; TEST IF ZERO
        RZ              ; IF ZERO THEN DONE
        INX     H       ; NOT DONE, SO POINT AT NEXT PARAMETER
        DCR     B       ; DECREMENT PARAMETER COUNT
CMD3:   IN      STAT73  ; READ 8273 STATUS
        ANI     CPBF    ; TEST CPBF BIT
        JNZ     CMD3    ; WAIT UNTIL CPBF IS 0
        MOV     A,M     ; GET PARAMETER FROM BUFFER
        OUT     PARM73  ; OUTPUT PARAMETER TO PARAMETER REG
        JMP     CMD2    ; CHECK IF MORE PARAMETERS


; RETURNS RESULT FROM 8273
; RETURNS WITH RESULT BYTE IN A
IMDRLT:
        IN      STAT73  ; READ 8273 STATUS
        ANI     CRBF    ; TEST IF RESULT REG READY
        JZ      IMDRLT  ; WAIT IF CRBF=0
        IN      RESL73  ; READ RESULT REGISTER
        RET             ; RETURN TO CALLER


        END
```

□

<u>Mailing List, Contact Persons</u>

In response to many requests, we will publish our mailing list in the next newsletter. This will help people to locate other packet radio enthusiasts near them.

If you do not wish your name and address to be published, let us know now.

If you wish to be known as a contact person in your area, let us know and we will publish the fact.

□

NETWORK ARCHITECTURE AND PROTOCOLS FOR A WIDE-SPREAD AMATEUR DIGITAL COMMUNICATION NETWORK

by

DOUGLAS LOCKHART, VE7APU   OCTOBER 1981

In the last couple of years the Vancouver Amateur Digital Communication Group programmable communication controller has been used in many areas of the U.S. and Canada. As one of those who worked on the development of the board and its software, I am very pleased to see that it has gained fairly widespread acceptance in the Amateur Radio fraternity. It was not so clear, a couple of years ago, whether or not it would be accepted since it involved the use of techniques unused in Amateur Radio at the time. My impression of Amateur Radio at that time was that it had a great deal of inertia or resistance to change. But at the same time, like a massive body, once it gets moving has a large momentum. Now I believe that Amateur Radio is moving into digital communicationsand that nothing is going to stop it. We only need to guide it to the best system that we can. And what is the best digital communications system for Amateur Radio? I don't think anyone knows. The design of a commercial digital communication network costs hundreds of millions of dollars. That's right! - just for the design, not for the implementation. Yet, even after all this expenditure, most commercial systems have their problems and detractors. So, in spite of the small amount of money that Amateur Radio will be spending on network design, we may still be able to come up with a system equal to or surpassing commercial designs. With this is mind I will outline the general philosophy of the system we are working on in the VADCG.

Firstly, we wanted a low-cost interface to the network for an end-user. We felt that a user should not need to have a computer just to access the network. For this reason, we designed, produced and programmed the VADCG programmable controller. Of course, there were many other good reasons for going this way, but I am mainly trying to show the function of the controller in the network.

The network we designed the board for was not intended to be homogeneous but a network in which nodes would have different functions. Some of the node functions identified were:

1. A 'Terminal' or 'End-user' node. Typically, someone with only a teletype or video terminal, although a user accessing the network through his microcomputer would also qualify in this category. (i.e. An intelligent terminal.)

2. A 'Gateway' node. A node which allows users on the network to access another communications system. Examples:

   2.1 A gateway to the telephone system using an auto-answer/autodial 103 type modem.
   2.2 A gateway to a digital communications channel on a satellite.
   2.3 A gateway to the local VHF RTTY channel.
   2.4 A gateway to another amateur digital communications network.

Note that if a node is used to interconnect two networks which have the same protocols, it should not be called a 'gateway' because, in this case, the two networks are actually only parts of a larger network.

3. A 'Repeater' node. Used to extend the coverage of the network.

4. A 'Logging' node. To record activity on the network to satisfy regulations as well as for performance analysis.

5. 'Host' node. This is the computer system attached to the network and is usually the system that the end-user wants to use. It contains the programs and files that the user wants to use, such as editors, games, compilers, assemblers, file transfer programs, files of swap and shop information and mailing lists,etc.

6. 'Station' node. Coordinates the operation of the other types of nodes in the network. Provides network services and communication between the network and the end-user, repeater, logging, gateway and host nodes. At present, all messages pass through the station node but this is not an absolute requirement in order for the station node to do its job. The station node provides the higher levels of network protocol that the simple end-user cannot provide for himself because of the limitations in storage capacity and complexity in the end-user interface.

7. 'Message-switching' node. This is a node which has sufficient storage capacity to be able to store messages and data for an extended period of time. Such a node would be something like a CBBS system. Information which could not be transmitted to its destination immediately could be left here to be sent onward when the destination node was available. The communication network is a packet switching network and so it has little storage capacity for messages. Messages are sent through the network only when both the source and destination nodes are available. The message switching node could have messages to be received by any user on request as well as messages intended only for a specific user.

As you can see, the station node has a much more complex task than any of the other node types. Furthermore, the station node becomes almost indispensable in a system designed to use it. Because of heavy reliance on this node, it should be backed up by another station node in the area or by a repeater node allowing communication to another area which also has a station node. The hardware for the station node should be fairly reliable since it involves no moving parts. The station node being used by the VADCG for example is a three-card S-100 bus system. One card is a standard CPU card, another is a 64K dynamic memory card - both of these are standard S-100 cards readily available from many suppliers. The third card is a special I/O card the VADCG has developed for handling the special needs of the station node. The card has four channels of HDLC communication using the Intel 8273 chip and six interval timers. The interval timers are used to handle line timeouts and to simulate a time-of-day clock. The timers and the HDLC channels are all interrupt driven using 16 channels of vectored interrupts provided by two AMD9519 chips. Also using the interrupt structure is the power failure circuitry, the transmitter fault detection circuitry and the circuitry to detect software failures or loops. A failure in any of these areas will cause the CPU to enter a program contained in up to 8K of EPROM storage on the same card. This program allows upline reloading of the station node software or downline dumping of the station node software for analysis of software errors. Each channel has a choice of Baud rates and can operate with either synchronous or asynchronous modems. A number of extra control lines for input or output are provided to control external devices.

Some of the functions and services provided by the station node are:

1. Establishment and termination of virtual connections between nodes in the network.

2. Communication with the end-user in plain language. For example, the station node will provide an explanation of why a virtual connection could not be made. It will interpret and act on network commands submitted through a terminal keyboard. It will provide a list of the status of other users signed onto the local domain or provide a list of users in another domain. (for example).

3. Drive a logging node to record the connection/disconnection of the users of the network giving times and dates as well as usage statistics.

4. Drive a repeater node so that the repeater will do intellignet repeating of frames. Not all frames received by the repeater should be repeated.

5. Provide the higher levels of protocol required for an extended network for the minimum end-user system.

6. Make routing decisions and keep dynamically, information on delay times. The routing system as planned will use a distributed delta routing system allowing multiple paths for communication between station nodes something like Arpanet routing scheme. Routing decisions will be based on minimal delay time. Changes in delay times detected by a station node will be passed to adjacent station nodes. New station nodes in the network will be integrated dynamically and will be deleted when communication is lost.

7. Communicate with non-end-user nodes in the network using concise coded or formatted network commands suitable for computer interpretation and generation.

The above is not a complete list of functions provided by the station node. Others will probably be incorporated as the system develops but this list should give the idea of what the function of the station node is.

It should be noted that the above six types of nodes are not the only types possible but only the ones which we have identified as being the most important at the present time. Most functions can be identified as belonging to one of these six types, even though there may be occasions where there is an overlapping of function. See Figure 1 which should help to clarify the relationship of the nodes. Each station node has a 'domain' associated with it. The domain is the set of nodes that the station is providing services for. The domain is typically a geographical area such as a city but different station nodes may operate on different frequencies in the same geographical area. Also, one station node may operate on different frequencies and different Baud rates at the same time. The lines between the nodes on Fig. 1 represent logical communication links at the datalink level of communication. Not all possible communication links are allowed. Direct communication is only allowed between a station node and another node type or between two station nodes. However, a repeater node may be used as an intermediate node in this communication
Any message sent between non-station nodes has to be routed through the station node in each domain. To some, this may appear as a harsh restriction on the communication possible for, after all, there may be nodes in the domain that can communicate directly because of their proximity. To answer this, let's look at the advantages of going through the station node and the reasons f[illegible]communication with the station node.

1. Standardization of the radio link. Each node's equipment only has to be set up to interface with one point. This means that adjustment to the modem, power of the transmitter, orientation of the antenna and various other requirements for establishing a communication link only have to be set up for one link, not requiring a large amount of coordination with various nodes. Once establishment of communication with the station node, no other concern for communication with the rest of the network is required.

2. Low power and directional antennar can be used for the link. No rotator is required even if using a directional antenna.

3. Nodes which are out of broadcast range anyway would have to go through the station node.

4. Nodes which were using a different band would have to go through the station node to communicate.

5. Nodes which were using different speeds would have to go through the station node.

6. Nodes which require protocol translation would have to go through the station node. (more on this later.)

7. Nodes communicating outside of the local station node's domain would likely have to go through the station node.

8. All nodes using network services would have to communicate with the station.

9. Establishment and termination of connections between nodes would have to be arranged through connection services in the station node.

The above considerations do not totally rule out the channel sharing advantages of being able to have nodes communicating directly on the same channel as that of the station node when they are using the same speed and not requiring protocol translation and are within communication range. The protocol would have to be more complicated to allow these two types of communication to be carried on the same channel and yet allow coordination by the station node. All I can say is that the present software does not coordinate communications on the channel which do not pass through the station node. The software, however does ignore all addresses which have not been assigned by the station node so that other digital communication can share the channel. It would probably be more appropriate that these nodes use another channel for their communication since they appear to have little need of the network.

You are probably wondering how the VADCG programmable communications controller fits into this network architecture since most users of the board are using software in the board which communicates directly from one end-user to another end-user. Well in fact, this software which is commonly in use was written after the original software for the station node architecture had been written and was already in use. The terminal-to-terminal software is actually a modification of the original software to get it to work in the station-less environment. In fact, the hardware was limited to 4K of EPROM and 4K of RAM because the higher levels of protocol were going to be provided by the station node or by the host node. In spite of the general usage of the board for direct communication, it is still the intent of the VADCG to develop a network based on the station node concept. More circuits have been developed and software has been written for use in this type of network recently.

As Figure 1 shows, this architecture is distributed at the station node level but not at the lower levels. There are multiple communication paths between station nodes, but only

single paths between the station node and other nodes in a domain.
The VADCG board can be used in the terminal node, the repeater node, the logging node, the gateway node and the host node. However it probably is not suitable for use in the station node due to limited memory and the fact that it is a single channel. With suitable programming, it could possibly be used as a type of front end processor for the station node. The VADCG is developing separate hardware for the station node.

PROTOCOL LAYERS

<u>THE PHYSICAL LAYER</u> - This is the lowest level. It details the characteristics of the physical communications interface between the system components. We are adhering closely to RS-232 standards in the use of connectors, pin assignments and voltage levels but, in addition to the RS-232 serial interface,we are providing a TTL level parallel interface and a 20ma. current loop interface in order to accomodate the widest possible choice of end-user equipment.
<u>THE DATA LINK LAYER</u> - This layer manages the error-free transmission of frames over communication links between nodes in the system. Most communication networks are using a system very close to the HDLC standard as is the system we are using. This protocol is the same as that being used in the VADCG programmable controller for direct communication now. Unlike IBM's SNA, which supports only an unbalanced version of HDLC, we are using a balanced version in which neither node at each end of the link operates in slave mode. Both nodes share packet transmission and recovery responsibility. When this layer receives a frame in error according to the frame check sequence contained within each frame, it requests the retransmission of that frame and all following frames. The reception of each frame is acknowledged, and if no acknowledgment is received, some transmission fault is assumed to have occurred and corrective action is taken. This is usually an additional request for acknowledgment. If additional requests for acknowledgment fail then the link is assumed to have failed and other corrective action is taken. The protocol requires positive acknowlegment only after every 7 packets. The establishment of the link uses an initial connection protocol (ICP) in which information is exchanged between the connecting node and the station node. The connecting node passes a description of itself to the station node which keeps it in a table for the duration of the connection. The station node passes an assigned data link address to the connecting node which is used by both the connecting node and the station node for the duration of the connection. The protocol is half-duplex, multipoint and uses a carrier sense technique (CSMA) to resolve contention on the radio channel and improve throughput. The contention protocol used by the station node is slightly different than that of the other nodes in order to give the station node an advantage when contending for use of the channel. This is done because all traffic in the domain must pass through the station node. The station node is working for all the other nodes.
<u>THE NETWORK LAYER</u> - This layer provides services which transport data through the network to its destination node. Messages that are transferred between domains in the network require a full network address flow control functions. This information is added to the beginning of the packet as another block of information creating what I call a type 2 packet. The packets coming from a simple end-user do not have this additional information and are in a type 1 format. These services are provided in the station node but may be provided by a multi-user Host node. The decision to support type 1 or type 2 packets by a host node is indicated at the time of initial connection. When type 2 packets are selected, no translation of packets is done by the station and the management of the destination and source address fields as well as management of the sequence number is left to the host node. See figures 2 and 3 for the layout of the packets.
The following is an explanation of Figure 4:
After receiving a packet is translated into a type 2 packet using tables kept in the station node. The packet may already be type 2 in which case this translation is not necessary. The packet is then analyzed to see what its destination is. If the packet is not for this domain, then it is routed back to data-link control. The router uses routing tables kept by higher layers to decide what link the data should be forwarded on. If the packet is for this domain then it is either for network services or for another node in this domain. If it is for another node in this domain it is translated to type 1 if necessary and passed to the data-link layer. If it is for network services, then it is checked to see that it originated from an end-user terminal. It it did, it means that the data has been typed in using English words and must be parsed and analyzed by Terminal Input Services before being passed to Network Services for action.

As a result of the commands received by Network Services, Network Services may have control messages of its own to send to various points in the network. These control messages use codes suitable for interpretation by a computer. If they are to be sent to another domain, then they are sent via the router to Datalink Control output. If they are for this domain and have to be interpreted by an end-user (Terminal), they are passed to Terminal Output services which translates the codes to suitable English language sentences. The packet format is translated to type 1 if necessary before being passed directly to Datalink Control output. This technique has a couple of advantages. First, since a knowledge of the details of the characteristics of the terminal is kept in the domain's station node, Terminal Output Services has all the information available to do fancy formatting of the message to the terminal. It knows the line length, whether the terminal supports lower case, highlighting, gotoxy, erase screen, etc. This is not known at the remote Network Service point. Secondly, the computer format is more compact than the form put out by Terminal Output Services and so is more efficient at utilizing the longer communication channels.

Note that for every command that can be entered in through a keyboard by an end-user, there is a corresponding coded command suitable for generation by a computer. Likewise, for every plain language response to a command, there is a coded (or formatted) response for a computer program. This means, for example, that if there is a file transfer program running in a Host computer the file transfer program can establish a virtual connection with another node using the network commands, transfer data accross the connection and terminate the connection without human intervention. Host nodes are capable of establishing multiple virtual connections at the same time.

DEVICE SUPPORT

As mentioned earlier, the station node receives and holds information on the configuration of each connected node. This information is passed to the station at initial connection. In the case of a terminal node, this information contains details of the device characteristics and addresses in the node. When a connection is established between an application program and a device, the application program can request the device characteristic information from the station node. On the basis of this information, the application program can decide how to communicate with this device or even if it is capable of communicating with it. For example, suppose a user tried to use a full screen editor program but only had a hard-copy ASR-33 terminal. The application program can send an error message to the user and disconnect. On the other hand, suppose the full-screen editor program found that it was communicating with a video display, then it would need to know how many lines and columns were in the display, whether lower case was supported, whether highlighting was supported, etc. The full screen editor would then be able to communicate with the video display efficiently. This exchange of information binds the device and the application program if successful. There will be commands available to the end-user to dynamically change the device characteristic information after connecting to the station node.

SUMMARY

I was hoping to be able to go into more detail on the routing, device support and packet formats in this paper but I realize that each of these ought to be the subject of separate papers.

The author feels that the station node concept of network development offers the most function for the least cost to the minimal end-user. The specialization of function in the system prevents the waste incurred by duplicating the same code in every node. As new functions and services become available, they are instantly available to all users of the network. The routing decisions are made at the station node level and the network is distributed at this level. This appears to be a reasonable tradeoff since the routing code is fairly complex and maintains a large amount of network information. Furthermore, there does not appear to be a simple distributed routing system in the literature that is workable for the low-cost end-user node. The many advantages that the station node offers appear to strongly outweigh the disadvantage of having to rely on it. In any case, we will have to rely on something if we are going to get our messages relayed accross the continent reliably and I am sure that Amateur Radio is going to have its own digital communications network operating accross the continent before very long.

FRAME LAYOUT

```
+-------------+---------------------+----------------+------+--------------+
! LINK HEADER ! TRANSMISSION HEADER ! PROCESS HEADER ! DATA ! LINK TRAILER !
+-------------+---------------------+----------------+------+--------------+
!   3 bytes   !     2 or 10 bytes         3 bytes      var  !   3 bytes    !
!             !                                             !              !
!             !----------------PACKET-----------------------!              !
!                                                                          !
!-----------------------------FRAME----------------------------------------!
!                                                                          !
```

The 'frame' is the block of information that is physically transmitted on a link in the network. Note that this is not the same as the 'packet' which is only part of the frame. The 'packet' is the information that is actually passed from one node to another node in the network. The 'link header' and 'link trailer' surround the packet and are used to convey the packet from one node to another. The Datalink Control layer of protocol manages the link and adds the datalink header and trailer to the packet before it is transmitted and analyzes and removes the header and trailer when a frame is received.

LINK HEADER AND TRAILER

```
+------+---------+---------+--------+---------------------+------+
! FLAG ! ADDRESS ! CONTROL ! PACKET ! FRAME CHECK SEQUENCE ! FLAG !
+------+---------+---------+--------+---------------------+------+
| 1byte|  1 byte | 1 byte  |  var   |       2 bytes       | 1byte|
```

FLAG bytes are used to separate frames. The flag byte is the binary sequence of 01111110 = hexadecimal '7E'. This sequence will not occur between frames because of a 'bit stuffing' technique used to transmit the data.

The ADDRESS field identifies the link address. Each physical link allowed in the network has its own address. It uniquely identifies the two end points or nodes that the link physically connects. The uniqueness only extends as far as the domain of the local station node which dynamically assigns these link addresses. Although HDLC protocol standards allow for up to two bytes of address we are only using one byte at the present time because it is felt that 256 different links would not likely be operating in one domain at the same time. The use of two bytes of datalink address will be considered after more work is done with the existing datalink protocol. Note that both end nodes use the same link address when communicating.

The CONTROL field is very similar to the HDLC or SDLC standards except that only a subset of the possible frame types is used. The control field identifies the type of frame. There are three types of frames:

1. Information frame. This is the frame which actually contains the packet. The layout and usage is the same as in SDLC
2. Supervisory frame. Only RR (Receive Ready) and RNR (Receive not Ready) are supported. REJ (Reject) is not being used since only half duplex links are being used. Extensions of this protocol to support full duplex links will use REJ supervisory frames. Supervisory frames are used to manage the link but do not directly pass packets.
3. Non-sequenced frames are used to handle special or exceptional conditions on the link such as link startup and termination and logical errors encountered on the link. Only four of the 32 different types are used at present.

The FRAME CHECK SEQUENCE (FCS) field is a type of check sum of all the bytes in the frame. It is used to verify that all the information in the frame was received correctly. It is the same as the HDLC standard.

I will not go into any more detail about the Link header and trailer because this protocol is very similar to the HDLC protocol and it has been described in some detail in a copy of the AMRAD newsletter recently. Furthermore, this protocol has been in use for some time now and has proven to be reliable and effective in providing flow control and data integrity accross the various links in the network. So let's look at the next higher level which has not been fully implemented - the packet level.

PACKET LAYOUT

The 'Packet' is the information in an information frame. It is sandwiched between the link header and link trailer.

```
+---------------------+----------------+-----------+
! TRANSMISSION HEADER ! PROCESS HEADER !  DATA     !
+---------------------+----------------+-----------+
   2 or 10 bytes           3 bytes       variable
```

The packet is divided into 3 fields:
The 'Transmission Header' contains information that is used to route the packet through the network and provide for the orderly flow of packets in the network. This information is no longer needed when the packet reaches its destination.
The 'Process Header' describes the data field to the destination program. The information contained in the process header allows for the orderly exchange of information between two processes. The concept of a 'process' may not be understood by many of the readers so I will digress for a moment and try to explain what is meant by a 'process'. In a network, communication is not done directly between devices at each end. The communication is actually between programs operating at each end. For example, let's say you were typing on the keyboard of a TTY connected to the network and the data was being printed on a printer a long way off connected to the same network. This is what it would look like:

```
+-------+  +---------+                                +-----------+  +---------+
!       !  !         !                                !           !  !         !
!  TTY  !--!   TTY   !____________Network_____________!  PRINTER  !__! PRINTER !
!       !  ! DRIVER  !                                !  DRIVER   !  !         !
+-------+  !         !                                !           !  +---------+
           +---------+                                +-----------+
```

As you can see, it is the two drivers that are exchanging the information passed from and to the network. The drivers could also be called processes.

i.e. the TTY process and the Printer process. 'Process' is a more general term than 'driver', which only implies a hardware device. A process can serve a program as well. Getting back to the above example again. The type of information exchanged between processes is very much different depending on the type of processes. The information in the process header is put in by the source process and stripped off and used by the destination process. It will not be passed directly to the printer in the above example. The TTY process may want to tell the printer process the tab settings that will be required on the printer. The information in the process header allows the printer driver to discriminate between data listing the tab settings and data to actually be forwarded directly to the printer.

TRANSMISSION HEADER LAYOUT

```
 0               1               2               3
 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
! Type ! Flags !   Reserved    !   Destination Address         !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!    cont.    !            Source Address                      !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!          Sequence             !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: 4 bits
    Indicates the format of the transmission header. The figure above describes a type 2 transmission header which is ten bytes long. and would have a value of 0010 binary in the type field. There is also a Type 1 transmission header which is only two bytes long and is used for simple end users of the network and allows the station node to provide the higher levels of transmission protocol. It will be described later.

Flags: 4 bits
    Controls segmentation of packets and expedited flow of packets.

```
  3    2    1    0
+----+----+----+----+
! Segment ! R  ! EF !
+----+----+----+----+
```

```
Segment                          R = reserved bit    EF =Expedited flow
  11  -  Only segment                                1 - Expedited
  10  -  First segment                               0 - Not expedited/
  01  -  Last segment                                    Normal flow
  00  -  Middle segment
```

Since there may be a requirement in the future to reduce the size of the packets to allow them to pass over links or through other systems which need smaller size packets to operate, segmentation (also called fragment-tation) of packets is supported. Packets may be reassembled again using the information in the segment field. The segmentation field is not used when the expedited flow indicator is on and expedited packets may not be segmented. The sequence field is not used when a packet is expedited as it will be handled out of sequence.

Destination Address: 24 bits
    The originating address. The final usage of all these bits has not been finalized. The first byte (8bits) is the originating domain address, the second byte is the link address in the domain and the third byte is used to identify the process operating at the end of the link. Note that the boundaries between these three addressing levels may be moved if equal addressing range is not required. This field could be also used as two levels instead of three when communicating with a domain without a separate station node.

Source Address: 24 bits
    The originating address. The layout and usage of these bits is the same as in the Destination Address field.

Sequence: 16 bits
    This is a wraparound count of the bytes passed between the source and destination. It is used to allocate buffer space when packets arrive out of sequence at the destination node. It is adjusted by segmentation routines when a packet is segmented. The segments of a packet may be recognized and space reserved for the segments which are still missing. When the missing segments arrive, they may be inserted in the buffer and the rebuilt packet may be passed along to the destination in the proper sequence.

TRANSMISSION HEADER LAYOUT (Type 1)

    The Type 1 Transmission Header is only two bytes long. This format of transmission header should only be used with a station node which will provide the higher level transmission protocol for nodes using Type 1 headers. Many simple end users will want to be able to use the network efficiently but may have insufficient resources to manage the complete network protocol which involves end-to-end flow control and packet reassembly and sequencing. The layout is as follows:

```
 0               1
 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!  Type ! Flags ! Process Addr. !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: 4 bits
    This is a type 1 Transmission Header and will be indicated by a value of 0001 binary in this field.

Flags: 4 bits

```
+----+----+----+----+
! NS ! Reserved! EF !
+----+----+----+----+
```

NS = Network Services
 1 - This packet is for network services or from network services
 0 - This packet is for a virtual connection
Note that if there is no connection known by the station node, then the packet will be sent to Network Services anyway.

EF = Expedited flow
 1 - Expedited data
 0 - Normal data

Note that the Type 1 transmission header does not support segmentation.

Process Address: 8 bits
    This will identify up to 256 source or destination processes running in the node. (It is unlikely that this many processes will ever be running in a node using type 1 transmission headers.) There is no ambiguity caused by the missing information since the node only receives packets from the station node so if the packet is on an output queue it indicates the source of the data and if it is on the node's input queue then it indicates the destination of the data. The station node also knows the domain address and the link addresses and so can build a type 2 packet from a type 1 packet.

PROCESS HEADER LAYOUT

```
 0               1               2
 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
!   FLAGS       !   RESERVED    !    USER       !
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flags: 8 bits

```
   7    6    5    4    3    2    1    0
+----+----+----+----+----+----+----+----+
! RR ! F  ! E  ! P  ! DR ! ER ! CHAIN   !
+----+----+----+----+----+----+----+----+
```

RR = Request/Response   Request = 0
                               Response = 1

This bit indicates whether or not the data is original or whether it has been sent as a response to a request. The meaning of the other bits are affected by this bit. If the packet is a response to a sequenced request packet then the sequence number of the request is used as the sequence number of the response to identify which packet this is a response to.

F - Formatted
1 indicates that the data in this packet is a network command. There are many different types of network commands and they will be detailed later.
0 indicates that the data in the packet is not a network command but ordinary data.

E - Error
1 = The data field contains error information
0 = There is no error information in the data field
This bit has no meaning in a Request - only in a Response
There are many different types of error responses and they will be detailed later.

P - Pacing
If this bit is on in a Request it means that the receiving process should send back an acknowledgment with this bit on in the response. The requestor will continue to send packets after this pacing bit has been sent but will stop sending and wait after a certain number of additional packets have been sent. This is a mechanism used so that a large number of packets will not be sent into the network when they cannot be delivered out of the network,yet at the same time, the requestor does not have to wait for an acknowledgment of every packet before he sends the next one. It allows packets to be sent quickly without clogging the network. The optimum number of packets to send before and after a pacing request depends on round trip delay time and other performance parameters.

202 EQUIV MODEM

The long awaited VADCG modem is going out to the board manufacturer now and will be available for shipping in about two months. Bob Livingston will be getting half of the 100 boards assembled and tested for us.
Prices will be Can. $15 for the bare board and $80 for the completed board.

TERMINAL NODE CONTROLLER
RS232 SERIAL KIT

A kit of parts including all IC's, the crystal, sockets and resistors, etc. is being assembled by the VADCG.
We are getting enough parts for 100 kits, although we won't be able to afford to keep more than 25 each of 8273 and 8250 on hand. Cost will be $130 Can. or U$112.
We will not include the bits necessary for 20 ma. loop as few people seem to use it and the parts should be easy to find. Likewise the parallel part and socket.
Deliveries permitting we will also attempt to provide separate supply of 8250's @ $15 and 8273's @ $50 Can.

LIP and TIP Programs on Diskettes or EPROMs

Currently we can supply the LIP and TIP programs, as listed in this newsletter, as follows.

1. On IBM 128-byte/sector soft-sectored 8" diskettes, both programs on one diskette, for $15.00 including postage.
2. On 2708 EPROMs. The LIP is on 2 EPROMs, as listed. The TIP is on one EPROM, and must be custom-burned. See the selection chart, which must accompany order, on page . Price, $10.00 for 1 EPROM, or $25.00 for all three.

Order form for Diskettes and EPROMs

What speed is the computer, or terminal, to be connected to the TNC? Circle one:
110, 300, 600, 1200, 2400, 4800, 9600, 19200, 38400 bps
Note: only speeds up to 9600 bps have been tested to date.

What data format does your terminal, or computer, require?
Check one: __8 data bits, no parity __7 data bits, even parity
__7 data bits, odd parity __7 data bits, mark parity
__7 data bits, space parity
Mark parity means setting the eighth bit to a permanent mark, or "1". Space parity means setting the eighth bit to a permanent space, or "0". If not sure, say 8 data bits, no parity.

What call sign do you want in the terminal node? (Up to seven characters.)

Are you connecting your TNC to a terminal? __ yes __ no
and/or a computer? __ yes __ no

If you are connecting to a computer, will your computer stop sending if the clear-to-send interface line drops? __yes __no

If yes, will the computer stop immediately, or only at the end of an output line? ____________________.

To:

VADCG, 818 Rondeau St., Coquitlam, B.C., Canada, V3J 5Z3

Enclosed is:

	Can.	U.S.	
____	$10	$10	for TIP EPROM only (see page 21)
____	20	17	for LIP EPROMs only
____	25	22	for all 3 programmed chips
____	15	15	for diskette with TIP and LIP programs.
____	32	30	for TNC board
____	130	112	for RS232 parts kit
____	15	15	for 8250
____	50	44	for 8273
____	15	15	for 202 radio modem card
____	80	70	for modem card, assm.+ tested.
____	15	15	for newsletter and membership___new,___renewal.
____	10	10	for newsletter only (>100km) ___new,___renewal.

Name________________________________Call________________________________

Address_____________________________City________________________________

Prov/State____________________Postal/ZIP code___________________________

Phone______________________Computer or Terminal________________________

(Please do NOT publish my name___address___phone no.___)